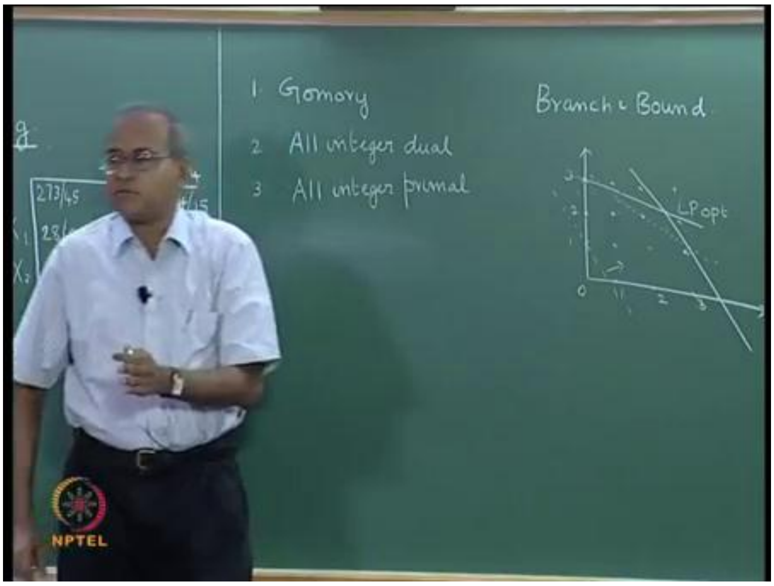
Advanced Operations Research
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

Lecture No. # 36
Integer Programming (Continued)

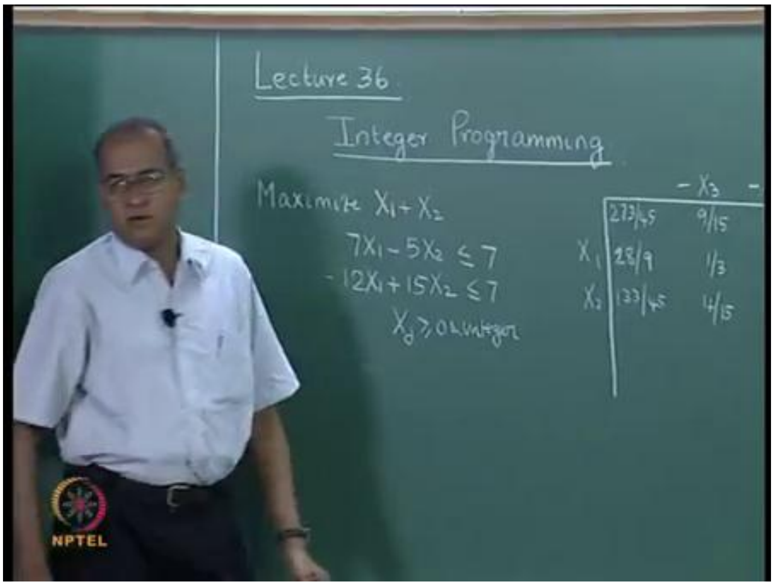
In this lecture, we re-visit integer programming. In the previous lecture, I had actually, wound up the course with a treatment on quadratic programming. Now, we are going to re-visit integer programming; largely due to some feedback that I have received and the need, to look into integer programming algorithms in little more detail.

(Refer Slide Time: 00:49)



In earlier lectures, we had seen three algorithms for integer programming. And these three algorithms are the Gomory cutting plane algorithm, the all integer dual algorithm, and the all integer primal algorithm. In addition to these three algorithms, we have also seen branch and bound algorithms to solve integer programming problems.

(Refer Slide Time: 01:40)



A typical integer programming problem would look like this; it will have a maximization or minimization objective; it will have a set of constraints; all decisions variables Xj will be greater than or equal to 0. And in addition, the decision variables also to integers. So when one or more of the decision variables take integer values, then it becomes an integer programming problem. And in this kind of, in this example both the variables X 1 and X 2 should take integer values; and therefore, it is an all integer programming problem, where all the variables have to take integer values. If we have situations, where some of the variables alone take integer values, while the rest of them can take continuous values, then such an integer programming problem is called a mixed integer programming problem.

In the earlier lectures, we have covered all the algorithms that we have covered so far - which are these three and the branch and bound - they have all been covered for the all integer programming problem, where all the variables were restricted to be integers. Now, in this lecture and in continuing lectures, we will also look at algorithms that address mixed integer programming problems in addition to addressing or re-visiting the all integer algorithms that we have already seen.

Now, if we see this classification, I have written the Gomory cutting plane algorithm, all integer dual and all integer primal algorithms on one side, and branch and bound on the other. The reason being these three come under a very generic category of what are called cutting plane algorithms, and branch and bound is a separate class of algorithms to solve integer programming problems. The essential difference lies in the fact that, at every iteration, we add one constraint, which is called a cut in a cutting plane algorithm, add a constraint to the existing solution, and then solve it again.

In a branch and bound algorithm, we essentially divide the problem into two by adding constraints of the type Xj greater than or equal to lj or Xj less than or equal to uj, creating two problems from an existing node and then finally, try to get the best solution among a set of problems that have been solved. Whereas, in the cutting plane algorithms, at the optimum, there is only one solution and that optimum solution is reached by adding cuts sequentially to an existing solution.

So, let us go back to these three types of cutting plane algorithms, and we also observe that, the cuts that we have seen earlier are different for each type of the algorithm. So, what we will do is, we will simply re-visit the problems that we have done, try and redefine the cut. And then try on address the question, is there a generic expression for a cut? Can we have something called a fundamental cut? From which each of this cuts can be derived. So, we will do a derivation for a fundamental cut, and then show that each of these cuts can be derived from the fundamental cut under certain problem conditions. And then we will also re-visit a few things as to the significance of how we get the all integer dual cut and the all integer primal cut.

So, let us go back to the example that we have already seen, we have used the same example in the earlier lecture, when we did the Gomory cutting plain algorithm as well as the all integer primal algorithm; we used a different example for all integer dual algorithms. The all integer primal algorithm essentially requires a maximization problem with less than or equal to type constraints and non-negative or positive values on the right hand side. So, such problems can be solved comfortably using an all integer primal algorithm. Irrespective of the objective function, whether it is maximization or minimization; irrespective of type of constraints, the Gomory cutting plane algorithm can be applied. So, we at use this example to explain both the Gomory cut as well as the all integer primal cut.

Now, before we actually get into the derivation of the Gomory cut, let us also try and understand how a cutting plane algorithm works in general. So, if we have a general integer programming problem let say, now let us say it is a integer programming problem with two variables X 1 and X 2; now let us assume that these are the integer points, let say the LP optimum is somewhere here; now, let us assume that LP optimum is here; please note that this is not the graphical representation for of this problem, this is the different problem. Here, I am just trying to draw a graph to explain a feasible region of a linear programming problem as well as an integer programming problem.

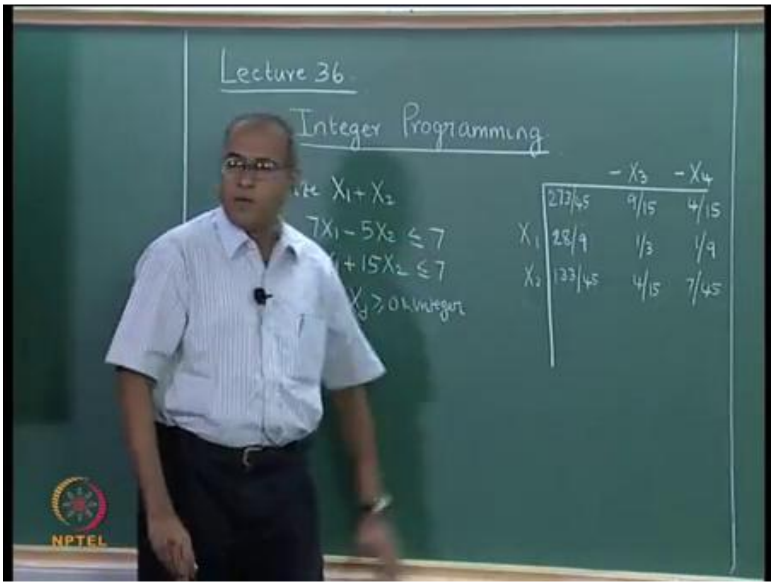
A feasible region of the integer programming problem contains only the set of feasible points, which are here; for example, a point here is not feasible to the I P assuming, this is 0, 1, 2, 3, 1, 2, 3 and so on. So, let say if the objective function is like this, and moves in the direction, this will be the LP optimum for this linear programming problem. And one of these will have to be plus of course, have to be the optimum solution to the integer programming. So, in a cutting plane algorithm, what we essentially try do is, to try and introduce constraints, which are in the form of cuts, such that these cuts or constraints remove a certain area from a feasible region, assuming two dimensions or assume certain space from the feasible region for higher dimensions.

And then at the end of it, when we continued to solve LP, at the end of it, the LP optimum will be an integer solution. So that is the fundamental idea behind a cut. For example, one could have a cut like this, if there is a cut which is like this, which takes away this space, and then when we solve a corresponding LP, now let us assume this will be the integer optimum solution. So, the purpose of the cut is to keep chopping of or keeping of removing areas from this such that at end of it, a corner point solution to an LP will be an integer point.

Now, we have already seen in earlier lectures that the cuts are slightly different for the Gomory cutting plane algorithm or the all integer dual algorithm and the all integer primal algorithm. But, essentially they do the same thing of trying to remove areas from this feasible region such that as you add these constraints and keep solving LPs; at some point, the LP optimum will be an integer solution, and it will also be integer optimum. Now, this also implies that the cut that we enforce or add into this does not eliminate or remove any integer point from the feasible region.

So, that is another thing that every cut has to ensure; the cut has to ensure that on addition of the cut, no integer feasible point becomes infeasible after the addition of the cut; the cut should satisfy all the integer feasible points in this region. So, when we generate a cut, we keep that in mind that no integer feasible point becomes infeasible with the addition of the cut. So, one of the rules when which have to follow is to ensure that the cut does not take away any integer feasible point from the feasible region; so, we will keep this in mind, and then we will re-visit how we created a Gomory cut from this.

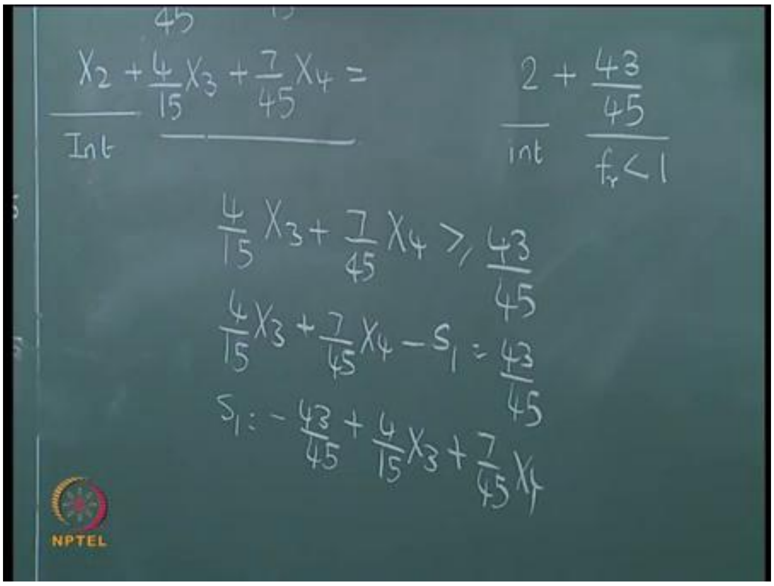
(Refer Slide Time: 11:06)



Now, let us go back to the problem that we have already solved using the Gomory cutting plane method. Now this is the LP optimum table that we have already seen in a previous lecture using a slightly different notation for the simplex algorithm; and we have already seen this notation also in the earlier lecture. So, this is the LP optimum to this problem, so the LP optimum has X 1 equal to 28 by 9, X 2 equal to 133 by 45 and the objective function value is 273 by 45 or 91 by 15. Now we, what we have done is we have solved the linear programming problem, which means we have left out this integer part, and we have just solved the problem up to X j greater than or equal to 0. We already know that if the LP optimum has integer valued solutions, then it is optimum to the integer programming problem as well. But, now the LP optimum is not feasible to the IP, because both X 1 and X 2 have fractional values, 28 by 9 and 133 by 45, which are not integer values.

So we have to now solve the integer programming by adding suitable cuts. So when we worked on the Gomory cut, we followed a simple principle whereby, we could either frame a cut based on this row or X 1 row or we could get a cut based on the X 2 row. Now, we followed a general guiding principle that we will generate a cut from a row or through a variable that has the largest fractional component. So the fractional component of this is 1 by 9, this is 3 plus 1 by 9, fractional component of this is 43 by 45, this is 2 plus 43 plus 45. So this has a larger fractional component and therefore, we said we will get a Gomory cut out of this. So, let me spent a couple of minutes, once again explaining the Gomory cut, so that we also understand the context of the Gomory cut, when we derive a general expression for a cut.

(Refer Slide Time: 13:28)



So, we go back and re-write this; we re-write this as, so we re-write this as X 2 equal to 133 by 45 minus 4 by 15 X 3 minus 7 by 45 X 4. So, this is the equation, which is represented by this. So, we re-write this equation in such a manner that the variables are on the left hand side and the constants are on the right hand side. So this would give as X 2 plus 4 by 15 X 3 plus 7 by 45 X 4 is equal to 133 by 45. Now the right hand side is written as an integer plus a positive fraction less than 1; this is always positive, so this is written as 2 plus 43 by 45. Now, because this is an all integer algorithm with all coefficients integer and the right hand side integers; X 1 and X 2 have to be integer valued at the optimum therefore, the slack variables X 3 and X 4 that we have introduced, which are here should also have integer values at the optimum.

Therefore, this is an integer, this is not an integer, this is not an integer, because we have a fraction multiplying X 3, and a fraction multiplying X 4. So, what we do now is, we keep this as it is. If there is an integer, we keep this. The coefficient here happens to be a fraction less than 1, so we retain it as it is. If the coefficient where a number, which is positive and greater than 1, then we write this as an integer plus a positive fraction less than 1; for example, if this where 24 by 15 X 3, this would have been written as X 3 plus 9 by 15 X 3. If this where 47 by 45 X 4, this will be written as 1 plus 2 by 45 X 4 or X 4 plus 2 by 45 X 4. Integers are written exactly as integers; now the next question is what happens, if you have negatives.

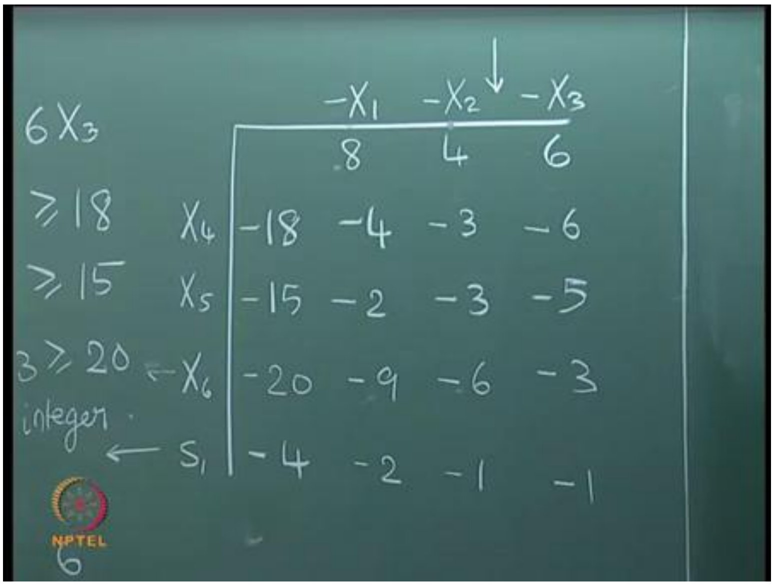
If this were minus 4 by 15 X 3, then this has to be written as an integer plus a positive fraction less than 1; so, if this where minus 4 by 15 X 3, this will be written as minus X 3 plus 11 by 15 X 3. If this where minus 7 by 45 X 4, this will be written as minus X 4 plus 38 by 45 X 4. So, this should always be written as an integer plus a positive fraction less than 1; it turns out there, there is the integer component 0, and the positive fraction less than 1 is 4 by 15 here, so it is written as 4 by 15; similarly, this is written as 7 by 45. For now what we do is, we keep the same thing again, and then we try and look at the left hand side as well as the right hand side.

So this is an integer, this is non integer, this is an integer, this is a fraction less than 1, strictly less than 1, fraction strictly less than 1. This is an equation therefore, at the end of it, this value should be equal to this value, which also means that this, at the end of it, if it has a fractional component, then that fractional component has to be equal to 43 by 45; at the same time, this is an integer, this plus this should have a fractional component of 43 by 45, but this can be 1 and 43 by 45 or even 2 and 43 by 45 it can be anything like that.

So, this portion alone can be an integer plus a positive fraction less than 1, and it will be it is, it can either be a positive fraction less than 1 or a positive integer and positive fraction less than 1, because this is a positive coefficient, this is a positive coefficient, these two have to be greater than or equal to 0 therefore, this quantity cannot be a negative quantity; the way we have written it; that is the reason we wrote this as a positive fraction less than 1. So this can be either 43 by 45 or 1 and 43 by 45 or 2 and 43 by 45 and so on. So we write this as 4 by 15 X 3 plus 7 by 45 X 4 is greater than or equal to 43 by 45, because this is either 43 by 45 or 1 and 43 by 45 and so on. So, this is essentially the Gomory cut. Now we make this as an equation by adding a negative slack, so this will become 4 by 15 X 3 plus 7 by 45 X 4 minus s 1 equal to 43 by 45; from which take s 1 to the other side so s 1 equal to minus 43 by 45 plus 4 by 15 X 3 plus 7 by 45 X 4.So this, on writing will become minus 43 by 45 minus 4 by 15 and minus 7 by 45. So, this is your Gomory cut.

A simple thumb rule to write the Gomory cut is when you are generating the cut out of this, take the fractional portion, put a negative sign, and write it minus 43 by 45. If this is positive, take the positive fraction less than 1 and write a minus, so positive take a positive fraction less than 1 and write a minus; for example, if this where 24 by 15, this will become minus 9 by 15; if this where 57 by 45, this will become minus 12 by 45. If this is negative, then do the complement in the sense if this is minus 4 by 15, then this will become minus 1 plus 11 by 15 therefore, this will become minus 11 by 15; if this where minus 7 by 45, it should be return as minus 1 plus 38 by 45 so this will become minus 38 by 45. So either one can go through these steps, and then write the Gomory cut or some we can simply write it this one. So, this is how the Gomory cut was written. So let us re-visit the all integer dual cut, which we have done earlier, we will go back and derive a general expression.

(Refer Slide Time: 21:41)



So, let us go back to the all integer dual cut, and see how we derive this first the all integer dual cut and that comes from, we consider a different example here, we have not used the same example in an earlier lecture, we have used the slightly different version of this example in the earlier lecture. So, we set up the initial table, so this will become we add X 4, X 5 and X 6; so we would start with X 4, X 5 and X 6, and we would have minus X 1 minus X 2 and minus X 3 here. This, we are going to use all integer dual algorithm; so an all integer dual algorithm implies that the dual is feasible and the primal is infeasible, so this will become 8, 4 and 6; dual is feasible, primal is infeasible; so minus 18 minus 4 minus 3 minus 6 minus 15 minus 2 minus 3 minus 5 and minus 20 minus 9 minus 6 and minus 3. Another way of doing it is one could say that 4 X 1 plus 3 X 2 plus 6 X 3 minus X 4 is equal to 18. So take X 4 to the other side, so X 4 is minus 18 plus 4 X 1 plus 3 X 2 plus 6 X 3, we already have a minus here, so you get all these things.

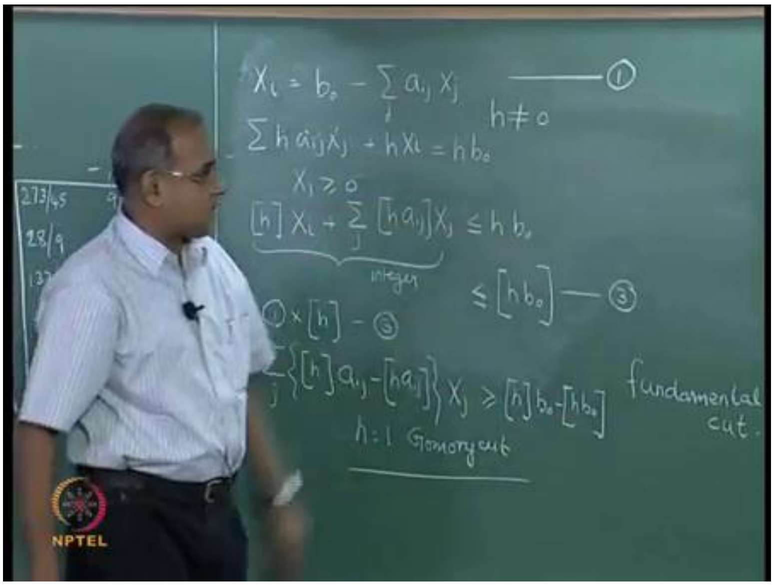
So, we go back and see, how we derive the all integer dual cut, what we did was, we started by looking at the variable that among the negatives, the one that is most negative was taken as a source row, so the cut is generated from this row, the X 6 row. So, if you re-visit the all integer dual algorithm, this is the source row, and then we look at all, how many of these have negatives? All 3 of them have negatives, so we go to the corresponding value 8, 4 and 6. And then the smallest one, we keep this 8, 4 and 6 here, the smallest amongst them is 4, so divide by 4 and take the lower integer value, so that would give as 2, 1 and 1.

Now, take the positive values of this 9, 6 and 3, find out the ratio, so you would get 2 by 9, 1 by 6 and 1 by 3. So, take the 3 values 8, 4 and 6, so we have the values 8, 4 and 6, let me write it here; the smallest among them is 4, so divide by 4, and take the lower integer value, so you get 2, 1 and 1; 1 comes because 6 by 4 has a lower integer value of 1. Now write the corresponding numbers here with a positive sign, so 9, 6 and 3; remember we have considered only those that are negative, so write them with a positive sign. Now divide them to get 2 by 9, 1 by 6, 1 by 3, and take the minimum of these, so 1 by 6 is smaller than 1 by 3, now 1 by 6 is 3 by 18, 2 by 9 is 4 by 18, so the minimum is it 1 by 6;

So take 1 by 6, multiply the source row with 1 by 6, take lower integer values and write the cut. So your cut will become S 1 variable; 20 by 6 or minus 20 by 6, lower integer value is minus 4 minus 9 by 6, lower integer value minus 2, minus 6 by 6 lower integer value is minus 1, minus 3 by 6 lower integer value is minus 1. So you have generated the all integer dual cut, and then you leave this variable out, this is the leaving variable, and this is the entering variable, pivot has to be minus 1 in an all integer dual algorithm, and these two are they have a pivot of minus 1; among all those that have a pivot of minus 1, pick the one which has the smallest value of the dual therefore, this is the entering column, this is the row; and one could do the iteration for the all integer dual algorithm.

So, now we have seen the way, two cuts are generated, we have seen the way, the Gomory cut is generated here for a different problem, we have also seen the way, the all integer dual cut is generated. So, now let us ask a question, is there a general way by which we can tie up all these cuts, and have a single formula or a single mechanism by which one can generate these cuts, so we try and solve that - you try and address that using a small derivation, which I am doing to do now.

(Refer Slide Time: 27:22)



So, let us assume that X i is equal to b naught minus sigma a ij X j summed over j. Now essentially, there is a source row from which we are generating the cut; if you look at a Gomory algorithm, this is the source row; the row that has the largest fractional value. If you look at the all integer algorithm, this is your source row, the one that has the most negative value amongst the infeasible values. So, the source row is written in this form in fact, when we wrote the Gomory cut that exactly what we did, we first wrote the source row. So this is a variable X i from which we are going to generate a cut, so in this case it will be X 2 equal to b naught is 133 by 45, the a ij X j or this 4 by 15 X 3 and 7 by 45 X 4.

So, you can write a source row this way, where b naught is your right hand side; the X i is a basic variable that is expected to be an integer, right now it is infeasible. Now, what we do from this is, multiplying by h not equal to 0, it is an equation, so we could have sigma h a ij X j plus h X i is equal to h b 0; so I am bring the a ij X j to the other side, right now h is not equal to 0, it is an equation, so h can even be negative, but later we are going to have only positive values of h. Now since all X j greater than or equal to 0, we can now write this as absolute value of h sigma a ij X j, this can be written as lower integer value of h into X i plus sum over j h a ij X j is less than or equal to h b naught. Now, let me explain this, this is not very difficult to follow, right hand side is the same, so the right hand side has not changed.

Now, I am taking this h out and writing the lower integer value, so this term can only be less than this term; similarly, I am taking the lower integer value of - this is the corresponding term, so this will be less than this; this is the corresponding term, so this will be less this. So the LHS here will be less than or equal to the LHS here; the right hand side is the same therefore, we convert this equation to inequality; so this will be less than or equal to h into b naught. Now this should be an integer, because all X i(s) and X j(s) are integers; this is the lower integer value of h, so this is an integer multiplied by another integer; similarly, this is an integer multiplied by another integer, so the left hand side is an integer.

And therefore, this has to be less than or equal to the lower integer value of h into b 0, this already holds, so all you do is take a lower integer value of this therefore, this will also hold. Now, we call this as equation 1, and we call this as equation 3. So equation 1 into lower integer value of h minus equation 3 would give us sigma…

(No audio from 31:54 to 32:24)

Now let us see how we get this expression. So multiplying the first one by h which means you are taking this to the other side, so multiplying this by - absolute lower integer value of h into a ij comes from here minus this comes from here, I am subtracting, so this part is fine. Similarly, I have lower integer value of h into b 0 coming from here, and h b 0 coming from the other one; and since I am subtracting one is an equation, and other is an inequality, so the sign gets reversed. So this will be greater than or equal to absolute value of h into b naught minus h b naught. Now this is called a fundamental cut. So depending on the values that you give to h, the cuts will change, and most importantly what we have taken care of is that every integer feasible solution will satisfy this cut; no integer feasible solution will be violated by this cut, by the fundamental cut; which is the very important aspect, when we which we understood when we were actually looking at the graphical representation of the problem.

So we have to ensure that all integer points satisfy the cut, and they will - the way the cut has been returned, they will satisfy. Now different values of h would give us different cuts, depending on the algorithm that we are using. So, now let us try and look at the Gomory cut from this; now when you look at the Gomory cut from this, you realize h is equal to 1 will give you the Gomory cut. How do we check that? The Gomory cut that we wrote from here to this, the Gomory cut is actually 4 by 15 X 3 plus 7 by 45 X 4 is greater than or equal to 43 by 45 is the Gomory cut.

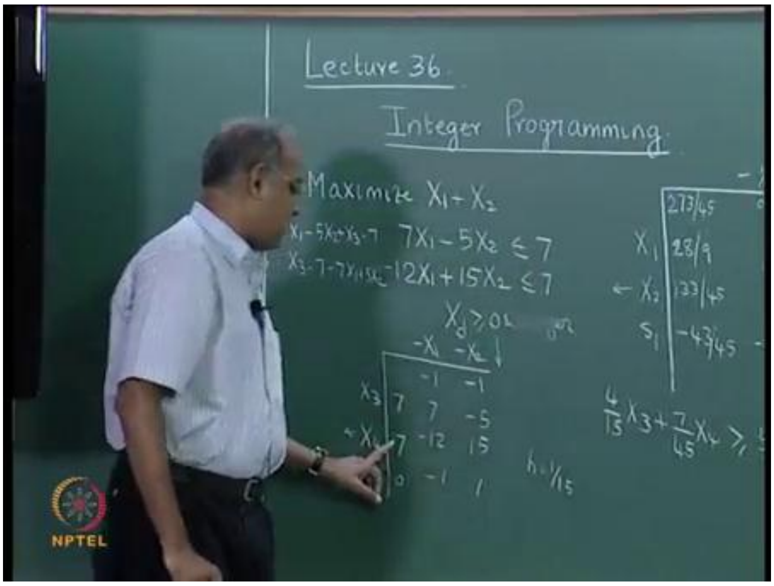
Now go back to this expression b naught is 133 by 45, so this is 1, so this is 133 by 45 minus lower integer value of 133 by 45, which is 2. So 133 by 45 minus 2 is 43 by 45, which comes here. Now go back to these, when you realize that this is a ij minus lower integer value of a ij, so the fractional portion was 4 by 15, so 4 by 15 minus lower integer value of 4 by 15 is 4 by 15; therefore, you got the 4 by 15. Similarly, 7 by 45 minus lower integer value of 7 by 45 is 7 by 45, because lower integer value is 0, so you got this.

So whenever that is why we use this, whenever this is a positive fraction less than 1 it automatically gets the negative sign. If this were 24 by 15, then it would have become 24 by 15 minus lower integer value of 24 by 15, which is 1 and 9 by 15, which is what we could have done with a minus sign; so this is the left hand side. If it were a negative, if this were minus 4 by 15, then this will become minus 4 by 15 minus lower integer value of minus 4 by 15, so this would have become minus 4 by 15 minus minus 1, so minus 4 by 15 plus 1 - 11 by 15, which would have come as a negative sign as 11 by 15. So the left hand side of the Gomory cut comes from this with h is equal to 1 would directly give us the Gomory cut now other values of h will suitably taken would give us other cuts.

Now if we actually observe this and see what we do here, we used in this case, h is equal to 1 by 6; if you remember right this is the value that we used. So let us try and substitute 1 by 6, and see whether we get this. Now what happens when we use h is equal to 1 by 6, lower integer value of 1 by 6 is 0; therefore, what happens? This will go, this will remain, this will go and this will remain; b naught was minus 20, so you got b naught is minus 20, you got minus of lower integer value of minus 20 by 6, I hope we are doing all right; b naught itself is minus 20, so minus 20 by 6 has a lower integer value of minus 4 with a minus sign would give us plus 4, which later on substitution would give us this minus 4; or when we move to the other side of the equation, you will get that value.

So, this does not exist, when h is a positive fraction less than 1 this term goes, this term also goes. Now similarly, you can see that in all of these the lower integer value of h into a ij is the one that remains. Now the a ij values are minus 9 minus 6 minus 3, h is equal to 1 by 6, so lower integer value on multiplication is exactly what we did, so minus 9 by 6 has a lower integer value of minus 2 minus 1 minus 1, and that is how the cut is. So the, in this particular example, the cut comes from using h is equal to 1 by 6. In a similar way, we can actually explain the all integer primal algorithm and the corresponding primal cut also I something that we can easily explained; which we can also do it for a moment, so if we were to apply the all integer primal algorithm to this, the table will look like this.

(Refer Slid Time: 39:36)



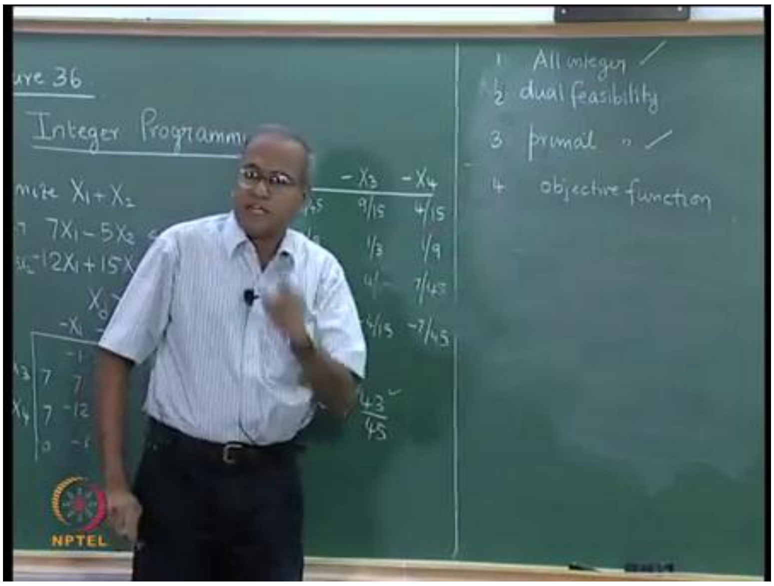
We would add X 3 and X 4, so we would start with X 3 and X 4 here, and we would have minus X 1 and minus X 2 here, this is the primal algorithm, so dual will be infeasible, primal will be feasible; so we would have got a minus 1 and minus 1 here, and then this will become plus 7 and plus 7 here. So X 3, so this will become 7 X 1 minus 5 X 2 plus this, so this will become, 7 X 1 minus 5 X 2 plus X 3 equal to 7, X 3 equal to 7 minus 7 X 1 plus 5 X 2. So therefore, this will become 7 and minus 5, this will become minus 12 and minus15. So if we applied the all integer primal algorithm, the most negative of this will enter first dual infeasible, so the most infeasible dual will enter; in this case, both are equal; so we could either start with X 1 or we could start with X 2.

So let us say we start with X 2; if we start with X 2, then we go back and say there is only one source, because there is a minus sign here, there is only one source, so this is your source, so your theta is 7 by 15, which is what you have. So what you do is 1 by 15 - multiply with 1 by 15 and take lower integer value. So this will become 7 by 15 lower integer value 0 minus 1 and plus 1. So in this case, h will become 1 by 15. Now once again, we can go back and show that h is a positive fraction less than 1 therefore, this will go, this will also go, so we will simply have lower integer value of h into b b naught. Now this is 7, 7 by 15 lower integer value is 0; similarly, h into a ij lower integer value, so there is a different h depending on the all integer primal algorithm, and another h depending on the all integer dual algorithm.

So this is how we can relate the three cuts that we have already seen to what is called a fundamental cut. So another aspect that has to be looked at is, this aspect that we did here, now this needs to be explain further. Even though we were able to get h equal to 1 by 6, and show that h equal to 1 by 6 when apply to this cut gives us this particular all integer dual cut. We certainly have to explain the little bit about why we take these numbers, why do we divide this by the smallest of these values, and then go back and divide by these fractions. So, that needs a little bit of explanation, and let us try and do that.

So, right now what I am going to do is, I am going to take out all of these. I am right now going to leave this, so we are only going to have this the original problem that is represented in the simplex table. Now, we need to add a cut, which is called an all integer dual cut. Now what do we want? Right now the way this table is written in the all integer dual algorithm, this is a minimization problem with all positive coefficients, all constraints of the greater than or equal to type; so perfectly amenable for the all integer dual algorithm. A simplex table will give you a dual feasible and primal infeasible solution, which is what is shown here; primal is infeasible, dual is feasible. Now assuming that we are solving LPs - Linear Programming Problems, what do we want? We want at the end, a simplex table where these are positive or non-negative, which means primal is feasible as well as dual is feasible. So any dual algorithm - number 1 is we want to maintain dual feasiblity.

(Refer Slide Time: 44:34)



So first one is any dual algorithm should maintain dual feasibility; so in fact even before that, it is an all integer algorithm, so we want to maintain the all integer nature of the algorithm. So let me say the first thing that we want to do is to maintain the all integer nature of the algorithm. Second thing that you want to do is to maintain dual feasibility. Third thing that you want to do the purpose of doing this is to ensure primal feasibility, at some point, some primal becomes feasible. Only then at the end at the optimum, we will have the optimum having both primal as well as dual feasible. So we need primal feasibility. And fourth of course, is we want the objective functions to increase or decreased depending on whether it is maximization or minimization.

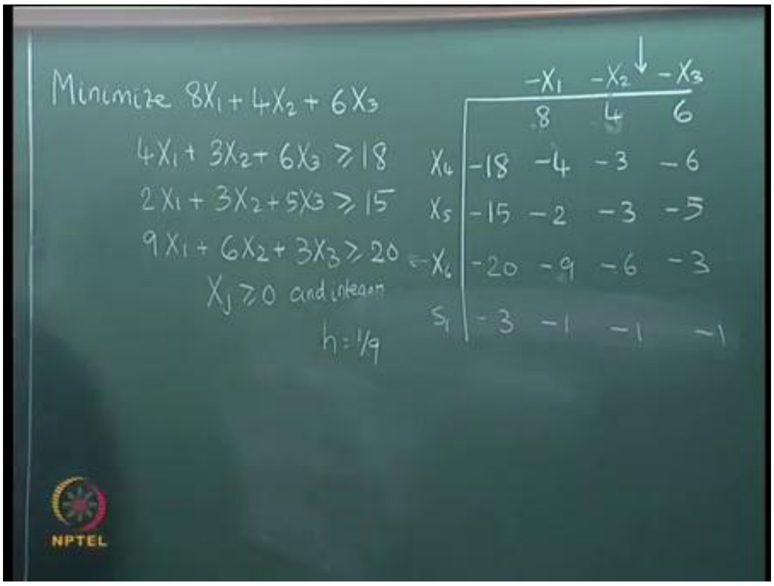
We want a large increase or decrease, so we are concerned about the objective function also. These are the four things that we want to do. Now the way this simplex table works, the way this particular simplex table works, now let us simply write this cut for a moment assuming h equal to 1 by 6, so this cut will be minus 4 minus 2 minus 1 minus 1. Now when we do an iteration of this, what are all the things that are going to happen? We also said this is the pivot, pivot row is divided by the pivot element therefore, this will become positive; pivot is minus 1; so when the pivot is minus 1, when you divide every element of the pivot row by the pivot, the integer character of that row is maintained. The pivot column becomes divided by the negative of the pivot, so the pivot itself is minus 1, so you are dividing it by plus 1 not only is this integer character is maintained, this one remains as positive, which is what you want.

So an all integer dual algorithm will have a minus 1 as a pivot. Having minus 1 as a pivot helps us in the following ways; one is this term will become positive, this term will remain as positive, and the all integer nature of the table will be maintained, because theonly place where you divide or this row and this column, every other place is only addition multiplication and subtraction and therefore, the integer characteristics will be maintained. So the pivot has to be minus 1 in a all integer dual algorithm.

So in an all integer dual algorithm, the all integer character is maintained when the pivot is minus 1; at least 1 of the primal variables becomes positive, when the pivot is minus 1. So now we have to do few things; we also need a pivot equal to minus 1, and then we have to find out that h, we have to find out that h; in this case, h is equal to 1 by 6 such that I get a pivot equal to minus 1 plus I have to maintain dual feasibility; now this minus 1 at the moment gave as feasibility here; it does not ensure the feasibility of this and this in the next iteration. So h has to be such that that feasibility is ensured, and h should also take as to a reasonably large increase or decrease in the objective function, so that we move towards to the optimum. So h has to be sufficiently chosen, so that all four are achieved. Right now what we have seen is a minus 1 pivot would help as comfortably achieve these two; now h should be chosen in such a manner that the pivot is minus 1 plus we also achieve these two.

Now, let us go back to this and see for example, what are the ways by which we can get a pivot of minus 1. Now a very simple thing is if h is equal to 1 by 9 or less than 1 by 9, we will get pivot equal to minus 1.

(Refer Slide Time: 49:41)



For example, if we do h is equal to 1 by 9, from these numbers the biggest of them is 9 the smallest of them is minus 9; so 1 by 9, h equal to 1 by 9 would definitely give as a minus 1 here. So if you use, let us use h is equal to 1 by 9 to get a cut, which is S minus 20 by 9, which is minus 3 minus 1 minus 1 minus 1; so this comes with h is equal to 1 by 9, but, please note that we have not used h is equal to 1 by 9, we have used h is equal to 1 by 6, because we did not use 1 by 9, we wrote 2 by 9 somewhere, the two came, because of this 8 by 4, and then we compare 2 by 9 with 1 by 6 and set 1 by 6. But right now 1 by 9 could also give us a cut. Now this cut is meaningful if it satisfies all 4 of them. Now we will check in the next lecture, whether h is equal to 1 by 9 is able to do that, and why we have chosen 1 by 6, instead of 1 by 9.

.