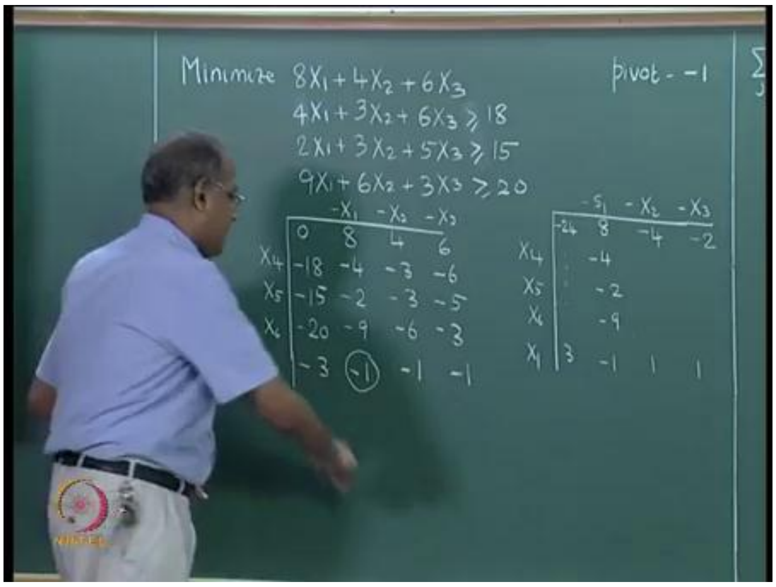
Advanced Operations Research

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

Lecture No. # 37
All Integer Dual Algorithm

(Refer Slide Time: 00:48)



In this lecture, we continue the discussion on the all integer dual algorithm. So, we use the same example that we saw in the earlier lecture to explain the algorithm further. So, this is the example that will be looking at, so minimization problem with all constraints greater than or equal to zero… As I mentioned in the earlier lecture, this example is different from a previous example that I have used to explain the all integer dual algorithm. We will proceed by taking this as an example.

So, we first set up the initial simplex table; the add 3 surplus variables X4, X 5, and X 6, which become the basic variables to begin with. So, we begin with X 4, X 5, X 6; the 3 decision variables X1, X 2, and X 3 are the current non basic variables. So, we write them as, minus X 1 minus X 2, and minus X 3. Now, it is a minimization problem, and it is a dual algorithm.

So, dual is feasible, primal is infeasible; so this is written as maximize, minus 8 X1 minus 4 X 2 minus 6 X 3 which would give 8, 4, and 6, here in the Z values. Now, we write this as X 4 equal to something; so there will be a minus X 4 here, so equal to 18; so X 4 goes to the other side, 18 comes to the this side. So, we get minus 18, minus 4, minus 3, minus 6. Similarly, minus 15, minus 2, minus 3, minus 5 and minus 20 with minus 9, minus 6 and minus 3.

So, we have set the first simplex - initial simplex table, and we now have to define the all integer dual cut. We saw in the previous lecture that the dual cut has to satisfy certain properties. First of all it has to satisfy the fundamental cut equation that we saw in the previous lecture.

So, we will now define or compute a h, and substitute that h into the fundamental cut, so that the fundamental cut equation or inequality is satisfied. Now, we wish a few things to happen: One is that we want to maintain the all integer nature of this table. So, if we wish to maintain an all integer nature of this table, then a pivot we have to have a cut in such a manner, that the pivot elements has to be plus 1 or minus 1. The reason is, the next iteration we are going to divide every element of the pivot row by the pivot element, and every column - every element of the pivot column by the negative of the pivot element.

So, that is only place, where we divide; in all other places it is multiplication, addition, and subtraction. So, if the original table has integers; the integer property of the table will be maintained. Now, we have to ensure that the division does not create fractions, and if the pivot is the plus 1 or a minus 1 then the all integer nature of this table will be retained. So, we wish to have a cut such that the pivot element is plus 1 or minus 1.

Now, in an all integer dual algorithm, if you assume that this is going to be the pivot column; so there will be a minus 1 here. And in the next iteration, the corresponding column will be divided by the negative of the pivot. So, if you have a plus 1 here, then the corresponding element in the next iteration will become minus, and if you have a minus 1 here, it will be plus. Since, we want to maintain the dual feasibility, we now have to have a pivot which is minus 1 and not plus 1. So, all integer dual algorithm will always have pivot equal to minus 1.

So, pivot is equal to minus 1 will ensure a couple of things: Number one - the minus 1 will retain the integer, character of this table. Number 2, it will also retain the dual feasibility of the corresponding column in a next iteration, and since every element of the pivot row is divided by the pivot, if there is negative here; then that negative will become positive, because you are dividing by minus 1. And therefore, one of the currently infeasible variables will also become feasible.

So, for all these reasons, it is good to have pivot equal to minus 1 in an all integer dual algorithm. But then, we also have to find out a h, which satisfies the fundamental cut, and results in a pivot which is equal to minus 1. So, in the previous lecture, we were looking at some possible values of h, so we first started by saying h equal to 1 by 9; if we do h is equal to 1 by 9, can definitely give us a pivot of minus 1. So, let us try h is equal to 1 by 9, and apply that h equal to 1 by 9 to the fundamental cut equation.

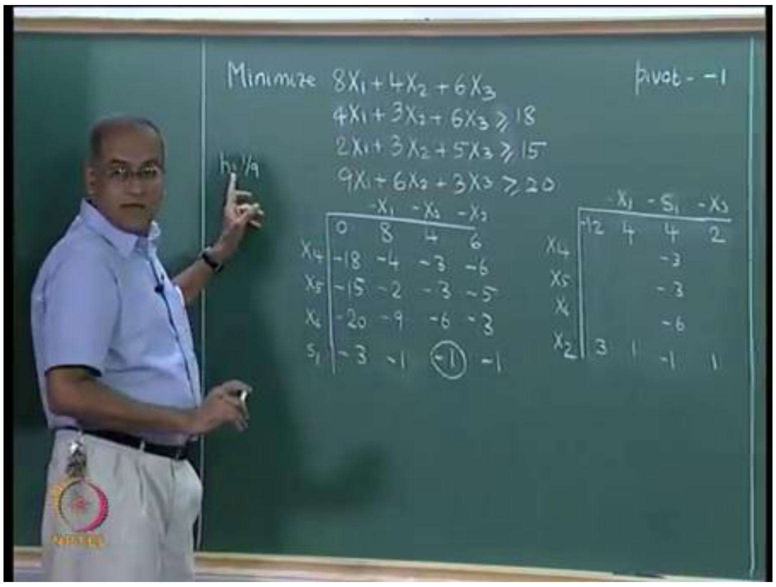
So, when we have h is equal to 1 by 9 our cut will become a slack variable S 1 will be introduced; so minus 20 by 9 lower integer value will be minus 3, minus 1, minus 1 and minus 1, because 9 into 1 by 9 lower integer value there is a minus sign, so this becomes a minus 1, 6 by 9 is minus 1, 3 by 9 is also minus 1. So, we can actually pivot on any one of this. Now, let us also try to understand which one out of these we can pivot, actually we have minus 1 here, minus 1 as well as minus 1. So, we could pivot on any one of these; so in some sense any one of them can be on entering column. So, let us first see whether what happens, if we pivot on this, and then understand why we should not do that, and why we should pivot only on this 4.

So, first let us try and check what happens, when we pivot on this 8. So, when we pivot on this 8, the next iteration X 1 will come here; so X 4, X 5, X 6, and X 1. So, I have minus S 1, minus X 2, and minus X 3; so I am going to assume that this is my pivot element. So, what will happen is I divide every the pivot element becomes 1 by pivot element; so minus 1 will become minus 1 here, divide every element of the pivot row by the pivot.

So, this becomes 3, this becomes 1, divide every element of the pivot column by the negative of the pivot. So, it will be retained as 8, minus 4, minus 2, and minus 9. This one will become, this is 0, at the moment; so 0 minus 8 into 3 will give me minus 24, and I can fill this column, but more importantly I am going to come here, so this element will become 4 minus 8 into 1 will give me minus 4. Similarly, 6 minus 8 into 1 will give me minus 2.

So, when I pivot on this 8, and when all of them have a minus 1; it is not advantages for me to pivot on a larger number here. Even though I could have chosen any one of them to be the pivot element; so automatically this makes as understand that amongst the minus 1 elements, we have to choose that column which has the smallest value here, so that we get the dual feasibility maintained. So, let me just show that also.

(Refer Slide Time: 09:11)



Minimize $8X_1 + 4X_2 + 6X_3$

$4X_1 + 3X_2 + 6X_3 \geq 18$

$2X_1 + 3X_2 + 5X_3 \geq 15$

$9X_1 + 6X_2 + 3X_3 \geq 20$

$h = 1/9$

pivot - -1

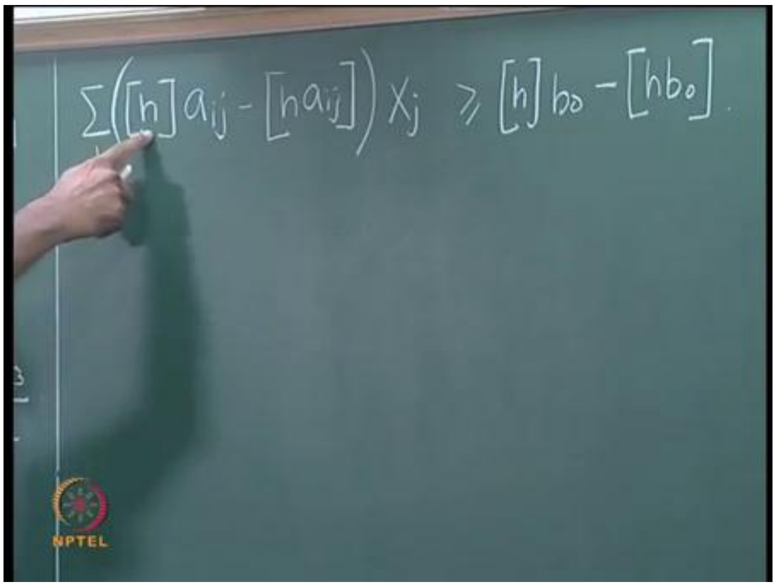
$$\begin{array}{c|cccc} & & -X_1 & -X_2 & -X_3 \\ \hline & 0 & 8 & 4 & 6 \\ X_4 & -18 & -4 & -3 & -6 \\ X_5 & -15 & -2 & -3 & -5 \\ X_6 & -20 & -9 & -6 & -3 \\ S_1 & -3 & -1 & \boxed{-1} & -1 \end{array}$$

$$\begin{array}{c|cccc} & & -X_1 & -S_1 & -X_3 \\ \hline & -12 & 4 & 4 & 2 \\ X_4 & & & -3 & \\ X_5 & & & -3 & \\ X_6 & & & -6 & \\ X_2 & 3 & 1 & -1 & 1 \end{array}$$

So let us try, and pivot on this; so this becomes the pivot. So, X 2 enters the solution, so X 1 remains here; and S 1 comes here. So, now this will become minus 1, 3 plus 1, divide by the negative of the pivot; so you get 4, minus 3, minus 3, minus 6. Now, this will become 0 minus 4 into 3 minus 12, and we can fill this; now this will become 8 minus 4 into 1, 4, and this will become 6 minus 4 into 1, 2. So, when I pivot on this I realize that the dual feasibility of this row is maintained. So I have to do two things: one is the pivot element has to be minus 1, and second is if I have more than 1 minus 1 element there, then I have to ensure that the entering column is the one that has the smallest value of this.

So, that is the second learning that we get. So, we can actually use h is equal to 1 by 9 to get this, to get the all integer dual cut, and then we can pivot on this or make X 2 as on entering variable, and then we proceed. So, it is perfectly allowed, and this is obtained by substituting h is equal to 1 by 9 in the fundamental cut equation.

(Refer Slide Time: 11:00)



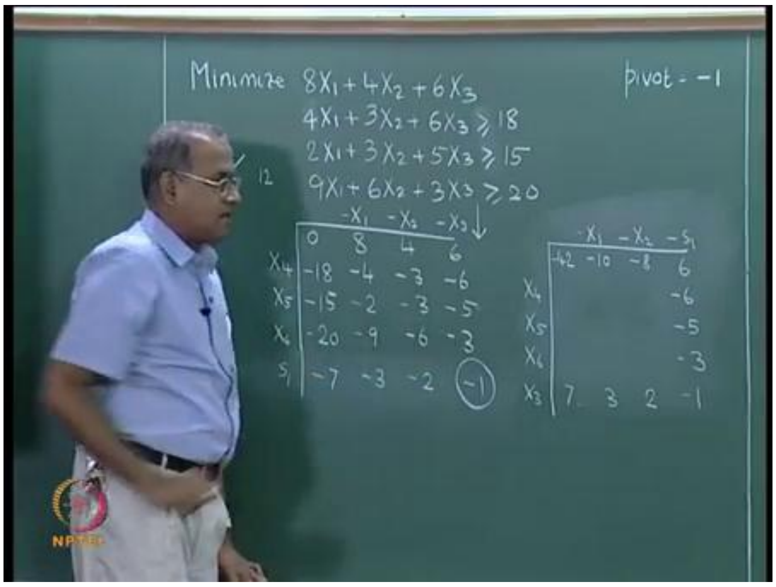
$$\sum\left([h]a_{ij} - [ha_{ij}]\right)x_j \geqslant [h]b_0 - [hb_0].$$

We have already seen in the previous lecture that this term - because h is equal to 1 by 9, the lower integer value of h will be 0. So, this term will go, so only this will be there; similarly, this term also will go. Now, what we have essentially done is to multiply with 1 by 9, and take the lower integer value which is what is shown here, as well as shown. So, for h equal to 1 by 9, we can write it. So, we could proceed with h is equal to 1 by 9, but then we go back and see in the previous lectures, as well as in the last lecture that I have use a slightly different way to compute h. And then we actually computed the h, we ended up having h is equal to 1 by 6, and not h is equal to 1 by 9.

So, let us now see this is one possibility, and I can use h equal to 1 by 9, and I can proceed. Now, let me look at some other ways of doing this. Now, let me go back to this equation, and see the other one, after all we choose h equal to 1 by 9 by choosing the largest of these numbers. And saying when I have h equal to 1 by 9, I will definitely get a minus 1 here, and a minus 1 here. Now, what happens when I get h is equal to 1 by 3, suppose I use h is equal to 1 by 3; I would still get a minus 1 here, but I will get some other numbers here.

So, when I get h is equal to… Let me try h is equal to, so I now say h equal to 1 by 9 is ok, I am able to get that. This is acceptable, because it is able to maintain the dual feasibility its able to do everything that I want to do, only difference is a gave me from 0 it move to 12 all right. The solution moves to 12, because lower integer value of 20 by 9 is in this case it is minus 3, so 4 into 3, 12 is the gain.

(Refer Slide Time: 12:43)



Minimize $8X_1 + 4X_2 + 6X_3$

$4X_1 + 3X_2 + 6X_3 \geq 18$

$2X_1 + 3X_2 + 5X_3 \geq 15$

$9X_1 + 6X_2 + 3X_3 \geq 20$

pivot = -1

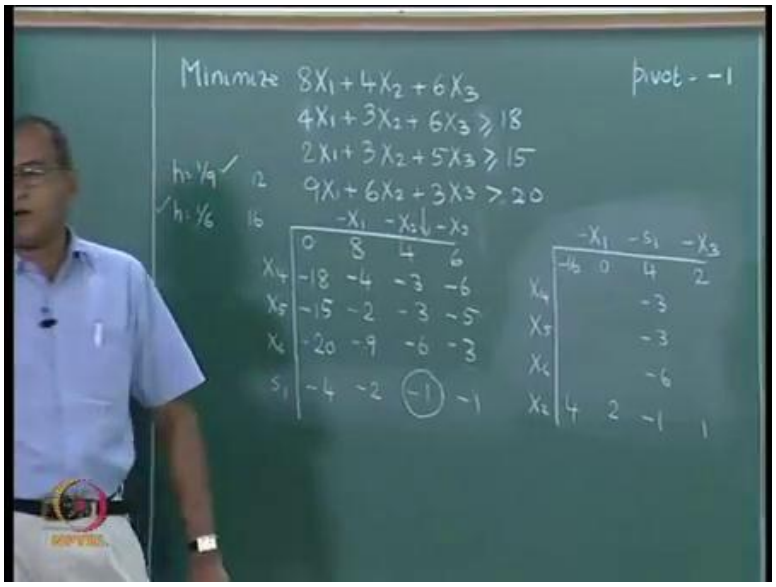
$$\begin{array}{c|cccc} & & -X_1 & -X_2 & -X_3 \downarrow \\ \hline & 0 & 8 & 4 & 6 \\ X_4 & -18 & -4 & -3 & -6 \\ X_5 & -15 & -2 & -3 & -5 \\ X_6 & -20 & -9 & -6 & -3 \\ S_1 & -7 & -3 & -2 & \textcircled{-1} \end{array} \qquad \begin{array}{c|cccc} & & -X_1 & -X_2 & -S_1 \\ \hline & -42 & -10 & -8 & 6 \\ X_4 & & & & -6 \\ X_5 & & & & -5 \\ X_6 & & & & -3 \\ X_3 & 7 & 3 & 2 & -1 \end{array}$$

Now, if I try h is equal to 1 by 3 my cut will look like this, I have S 1here. So, 1 by 3 will give me minus 7, because 20 by 3 lower integer value is minus 7; this will give me minus 3, this will give me minus 2, this will give me minus 1. And I know that I have to pivot only on the minus 1; therefore, this automatically forces me to have this as the entering column. We already know that this is not a very comfortable situation, but let us see what happens. So, when I do this again I realize that, I have minus X 1, minus X 2, minus S 1, X 3 entries the solution, X 4, X 5, X 6, X 3.

So, this is my pivot element; so divide - pivot becomes 1 by pivot, so I get a minus 1. So, this becomes 7, this becomes 3, this becomes 2, this is divided by the negative of the pivot. So, the same column remains 6, minus 6, minus 5, and minus 3. Now, this value will be 0 minus 6 into 7 - minus 42, and I can fill the rest of them; this value will become 8 8 minus 6 into 3, 18 which is minus 10. And this value will become 4 minus 6 into 2 which is minus 8.

So, pivoting on h equal to 1 by 3, which if we look at this we pivoted on 1 by 9, and set it as acceptable, but pivoting on 1 by 3 is not acceptable, because it we lose the dual feasibility of this which we do not want; it is a dual algorithm we want to maintain the dual feasibility of this. So, we will not pivot on 1 by 3 here.

(Refer Slide Time: 15:29)



The third and the last alternative is to try, and pivot on 1 by 6; so try and pivot on 1 by 6. So, if we pivot on 1 by 6 the constraint or the cut will become -20 by 6 is minus 4, -9 by 6 is minus 2, minus 1, minus 1. So 9 by 6 lower integer value is minus 2, 6 by 6 is minus 1 3 by 6 is minus 1. We already know that between the 2 minus 1 or amongst the several minus 1 we have; to choose the entering column as the one that has a smallest value. So, we could do this; X 2 enters the solution. And when we do this, we also realize that, so you have minus X 1, minus S 1, minus X 3; you have X 4, X 5, X 6, and X 2. So, this is our pivot; so one second pivot becomes 1 by pivot, so minus 1; divide by the pivot element 4, 2, 1; divide by the negative of the pivot; so 4, minus 3, minus 3, minus 6.

So, this will become 0 minus 4 into 4 - minus 16, and I can fill the rest of this. This will become 8 minus 4 into 2, 0; 6 minus 4 into 1, 2. So, at h is equal to 1 by 6, I can proceed just as I can proceed with h is equal to 1 by 9; it is still dual feasible, it does not become negative. One of them has become 0 will talk about it, but it is still not become negative. So, h is equal to 1 by 6 is acceptable as far as we are concerned.

So, we could either pivot on h equal to 1 by 9 or we could use h equal to 1 by 6, and generated all integer dual cut; the cut becomes different when you use h is equal to 1 by 9, and the cut become different when you use h equal to 1 by 6, because when you used 1 by 9 we had a minus 3 here. Which gave as an objective function of 4 into 3, 12; whereas, when we use h equal to 1 by 6, we are getting a minus 4 here, so this gives as 4 into 4 16.

One can also understand this 12 and 16 in a slightly different way. For example, this is like here c j minus z j or z j minus c j; this ratio is your theta or minimum ratio, because the pivot is minus 1, this will be straight away this product 4 by 1 is 4, 4 into 4 is 16. We also know in linear programming that the increase or decrease the objective function is the product of the minimum ratio, and the corresponding c j minus z j or z j minus c j with a sign taken appropriately. So, this gives as a 16, the other one gives as a 12.

Now, the question is would we prefer to use 1 by 9, and create an all integer dual cut which takes the objective function from 0 to 12 first or would we use a 1 by 6 to takes as to an objective function of 16. So, we go back to very convectional simplex method, we had always used what is called the largest coefficient rule, but this is something like your largest increase rule, because from 0 we have now move to 12, and we move to 16. So, it is advantageous.

Now, to look at h is equal to 1 by 6; so that we get into that 16 or other than we get into 12, because it is like using the largest increase rule in linear programming. So, compare to 1 by 9 we could now pivot on 1 by 6, and then we need to understand a couple of things. Now, this points we have understood that 1 by 6 as the slight advantage not saying great advantage, but a slight advantage; that it is able to give or reduce from 0 to 16 remember it is a minimization problem.

So, it goes like this at the end it will get adjusted, so 16 is preferred to 12. Now, we have to answer on other question, so between 1 by 9, and 1 by 6 we would choose 1 by 6, but then every time we cannot be doing all these substitutions like, we have done 3 times - we have done 3 iterations: One with 1 by 3, one with 1 by 6, one with 1 by 9, and then we understood that 1 by 3 is not going to help we does anywhere. Whereas, 1 by 9 and 1 by 6 are ok; we cannot do this every iteration trying to do it 3 times or 4 times depending on the number of negatives that we have. So, we need to have a formula or a computation through which we find the best value of h.

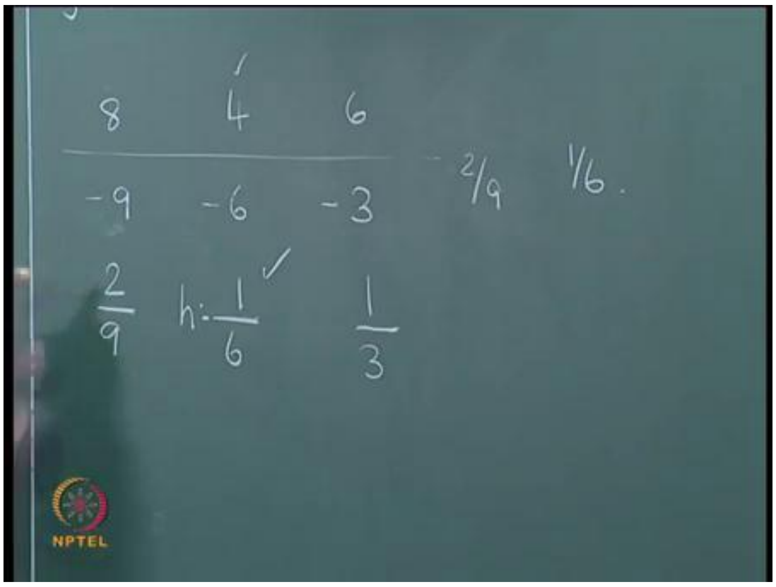
So, a simpler computation would be that computation should also be able to do two things; if we take simply these are the negatives. And therefore, we could say 1 by 9, 1 by 6, 1 by 3. And, suppose we take the smallest of them, as a thumb rule that would lead as to 1 by 9 with this. Whereas, we go that we have a way by which we could actually do this, and get a 16 which is preferable compare to pivoting on this, and getting a 1 by 9 for h, and getting up to 12.

So, our computation should eventually lead as to that, and it should be able to say or give a h such that going beyond that h - 1 by 6 in this example will give as infeasibility, which is what was shown by this 1 by 3. At the same time that value should be a limiting value, which can give as the so called maximum increase of this 16. We should be able to compute that kind of a number, how do we get that? Now, we get that, because the first thing we could do is you realize here that this 0 came, because of this two. If you go back and see how we computed this.

We this is our entering column, so the pivot column; so we wrote this first. This is our pivot row, we wrote this 4, 2, 1, and 1, we wrote this. And then, we said that this will become this minus this into the corresponding number here; so 8 minus 4 into 2. As long as this element is 0 or 1 or 2 this will become non negative; if this element where 3 then the corresponding thing will become 8 minus 4 into 3, which will become negative. So, we understand that as long as this number which is two, it was incidental there it is 2 is actually less than or is an integer less than or equal to this divide by this; that is how that two comes. It is incidental that this two also came, because of you know when we used a 1 by 6 minus 9 by 6; I am not looking at that, I am trying to say that their binding ratio is actually two, because this is anyway going to have a value of minus 1 here.

So, we have to ensure that the kind of h that we use to get this, should give as a value of a maximum value of two, such that the resultant number that we get here is non-negative. And that number is the fraction or the lower integer value of this and this which is two. Therefore, we go back, and define that when we generate a source use this as a source row for the all integer dual cut, we now go back and see which are all the negative elements there.

(Refer Slide Time: 25:08)



So the negative elements are minus 9, minus 6 and minus 3. Then we see what are the objective function or what are the z values; which are 8, 4 and 6. Now, we will also realize that if you take these are negative values, let us assume these are positive integers, these are positive integers and these are negative values. Simply look at these numbers; take the smallest of this, which is 4; and use 1 by 6, that is take this number, which happens to be minus 6, and use 1 by 6; let say we have that, is one number that we can think off.

Now, we will go back and say that, for this to be the highest and the best value, and for it to retain the dual feasibility. Then what should happen is that this ratio is 2, so 2 by 9 is the fraction here, so we compute a 2 by 9; and then say if 1 by 6 is smaller than 2 by 9, I will accept the 1 by 6; if this computed number is smaller, then I will use that h; I will use that h; and when I use that h, whatever is that h that comes here, when I divide it by itself, the pivot will always be minus 1. So I will be assured of a pivot element of minus 1, but if I look at these numbers, the fact that the ratio is 2, and 2 by 9 for h is bigger than 1 by 6; so 1 by 6 will give me a better value than 1 by 9.

My other alternative is to use 1 by 9, I said 1 by 9 would work, because 9 is bigger than 6, so when I use h equal to 1 by 9, I am going to get a minus 1 here, and a minus 1 here, because 9 is bigger than 6. And I can always enter this, so where dual feasibility is maintained, but how do I know that 1 by 6 can give me a better value than using 1 by 9 is by comparing this ratio, and getting this 2 by 9, which again as I said is the number that is going to come here at the end of the cut. So the number that is going to come here at the end of the cut as long as it is two or something, I am okay, and that is the two we are talking about which is the ratio. So if 2 by 9 is bigger than 1 by 6, then 1 by 6 can provide a slightly better value of h, which can give us a slightly higher number here, which is 1 by 6.

Therefore, what we do is, we take all those negative numbers here, then write only the negatives we take, we do not take the positive is easy to show that if one these number is actually positive, it is not going to affect the dual feasibility in any way; because if the number is positive, then the corresponding number here will become negative and therefore, we actually end up adding; we end up adding to these positive number, we do not end up subtracting. We subtracted, because these numbers are positive; if in the original case, these numbers also become positive, because these numbers are negative; you have to look at it very carefully, all the three numbers are negative, h is a positive fraction less than 1 greater than 0 therefore, all the lower integer values are negatives, negative integers.

And when we do the row operation, because these are negative integers other than this element all of them become positive integers. And when you have positive integers here, you actually end up adding something to this, so dual feasibility is not affected. So dual feasibility is not affected, when I have a numbers that is positive here and not negative; therefore, we take only the negatives to do this, because those negatives are essentially at the end of the day going to give as positive, which we end of subtracting. If some of these numbers are positive, then these numbers will become positive, and then these numbers will become negative, and then it means we end up adding something to this. Adding a positive quantity to a positive quantity is only going to retain the dual feasibility. So we are concerned only about those numbers which are negative here; so we take only those numbers, in this case all three numbers are there.

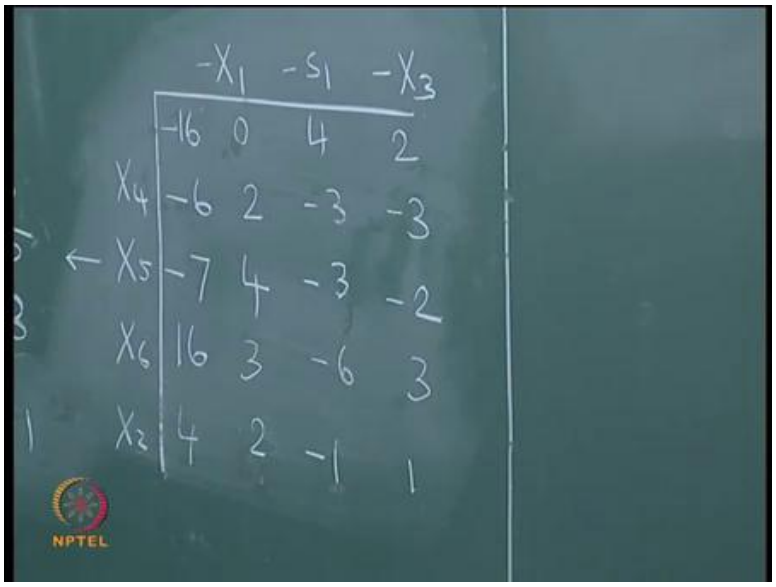
And then we realize that the effect of the impact of each of these is essentially, the lower integer value of the ratio of that number to the smallest number. It is always possible to take the smallest number here, and use one by that, and do a pivot, dual feasibility will be maintained. But then if we want a slightly higher value or the best value, so the impact of that is actually the lower integer value of the ratios therefore, we take that number, take this divide by this smallest value and take the lower integer to give us 2, 1, 1; and then you have this 9, 6 and 3; and then take the smallest of the 3 numbers. So 1 by 6, 1 by 3, 2 by 9, we take the smallest of these, and say that h is equal to 1 by 6 as the value that we will use to generate the cut; because any value greater than that will give us dual infeasibility in the next iteration.

A value smaller than that is okay, but at the same time since, we are interested in as large a value in the next iteration, we will take the corresponding value of h as the smallest of the three values. If you had not use this thing, we would have ended up with 1 by 9, instead of 2 by 9, and then we would have taken the minimum of 1 by 9, 1 by 6, 1 by 3; we could have used h equal to 1 by 9, we would have proceeded to another cut and another table with value going from 0 to 12. It will still lead us to the optimum solution except that it, could not saying it well, it could take more iterations than what this will take.

The approach that we use by doing this is equivalent of the larger increase rule and therefore, we increase from 0 to 16, instead of going from 0 to 12. So any value this is the like the best value, so any value less than this would also give us that as I said it is good to get this 1 by 6, and not pivot on 1 by 9, so that in principle or an average, we may get into situations where with fewer iterations, we will be able to solve this. So, that is how we get this value we choose between 2 by 9, 1 by 6 and 1 by 3; one has to understand how we get this 2; as I said earlier, this 2 comes from the lower integer portion of this 8 by 4, and as I explained here that 2 exactly constant to 0; otherwise it will get some other number and therefore, the whole thing will be very different, so we take the lower integer value of this.

So, we have to answer one more question; what happens for example, in the next iteration, when we do this, now this I will have some other value at the end of the iteration. Now what happens if one of the value is 0, like in this; so if one of the value is 0, we actually do not have any issue, we can simply take that particular value here and use 1 by that for h. Just to explain that let me complete this table, and then show how the next cut is generated.

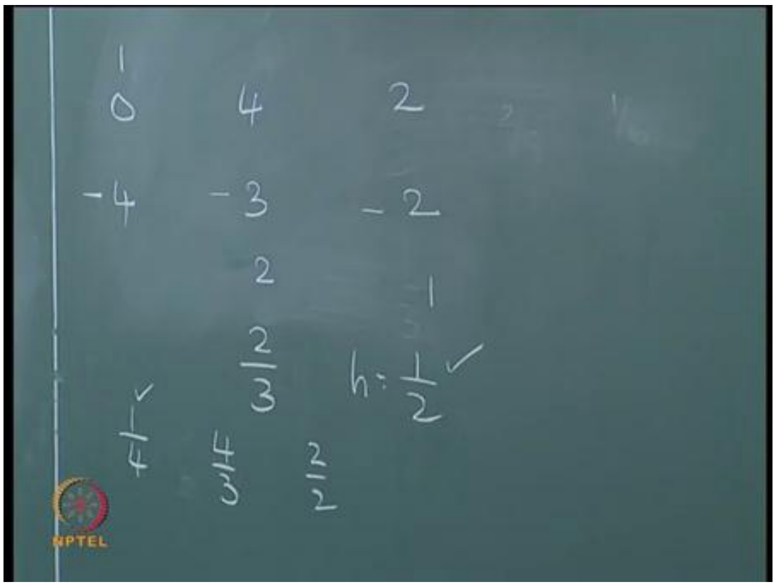
(Refer Slide Time: 35:00)



$$
\begin{array}{c|cccc}
 & & -X_1 & -S_1 & -X_3 \\
\hline
 & -16 & 0 & 4 & 2 \\
X_4 & -6 & 2 & -3 & -3 \\
\leftarrow X_5 & -7 & 4 & -3 & -2 \\
X_6 & 16 & 3 & -6 & 3 \\
X_2 & 4 & 2 & -1 & 1
\end{array}
$$

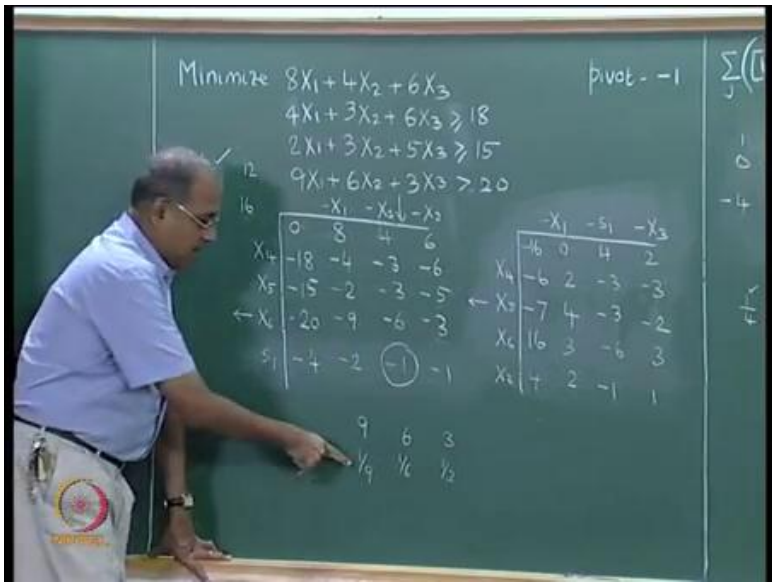
So we have to do this, so this is 0 minus 4 into 4 minus - 16 minus, 18 plus 12 is minus 6, minus 15 plus 8 - minus 7, minus 20 plus 36 is 16, this is 8 minus 4 into 2 - 0, minus 4 plus 6 - 2, minus 2 plus 6 - 4, minus 9 plus 12 is 3, this is 6 minus 4 into 1 - 2, minus 6 plus 3 - minus 3, minus 5 plus 3 - minus 2, minus 3 plus 6 - plus 3. So, this is the next table. Now we want to proceed from this table, we once again take the most negative one, which happens to be here, because the minus 7 is the most negative of the numbers, but then when we do this, we are having only two negative numbers, which are minus 3 and minus 2 we leave this out, because there is a positive number here, and because we leave this out we also leave out this 0.

(Refer Slide Time: 36:07)



So, we take only this 4 and 2; so the lower integer values become s1 and 2; so you have 2 by 3, 1 by 2, so h is equal to 1 by 2 we will use to generate the next cut. If for some reason, this was minus 4, instead of plus 4, then we would have taken that minus 4 also here; but then if we take a 0 here, then the lower integer value becomes difficult to compute, because 4 divided by 0, 2 divided by 0; in such a case what we in do is, we simply treat it notionally as 1, and then say 1 by 4, 4 by 3 and 2 by 2; so 1 by 4 will be the value. It is also not very difficult to understand that we are actually not making a mistake.

(Refer Slide Time: 37:23)



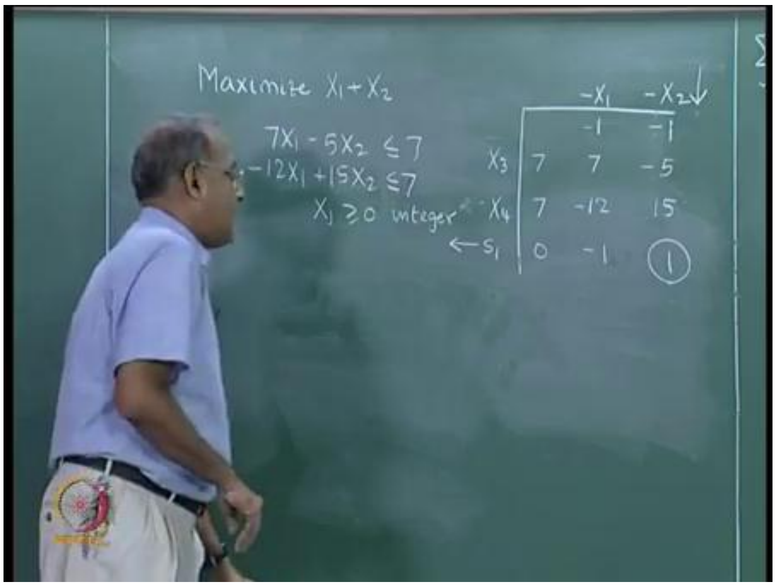
If you see carefully when we did the very first iteration, one again let me go back and explain, when we did the very first iteration, the values are 9, 6 and 3; h could have been 1 by 9, 1 by 6 or 1 by 3, the smallest of them could be 1 by 9; but then we said we want to make it better therefore, we took this ratio and make this 2 by 9. So the ratio actually helped us an increasing this numerator from 1 to 2, so that we got a slightly better value as the minimum; otherwise 1 by 9 would have been the minimum.

So the dividing it and getting the lower integer value is only a very notional way of understanding this kind of thing. So when we have a 0 here, one could easily take this as 1 by 4, h is actually 1 by that the number is enough. But we are trying to get a slightly bigger value of the minimum by doing this division, and putting it into the numerator. So, when we have a 0 there, you could treat as 1, for the sake of computation of h, and 1 by that number will automatically become h; of course, in this example it is not applicable, because we have a positive number; we would take only this negatives, as I said this will not come.

For this particular example, h will be equal to 1 by 2, and we can proceed. I was looking at a case when we had a negative here as well as a 0 here, then 1 by something, 4 by something, 2 by something. So this 1 by something will be smaller and therefore, and it will give the corresponding value of h. So this is how the basic idea of computing h and understanding and getting the right value h is done in the all integer dual algorithm with a with a guiding principle that we are trying to follow the largest increase rule by taking the appropriate value of h, which can give us the largest increase; at the same time do several things retain the integer characteristics on the table, have a pivot equal to minus 1, bring in a feasibility in the next iteration, retain the dual feasibility and try to give as large an increase in the objective. So this is how the all integer dual algorithm essentially works.

(No audio from 39:50 to 40:12)

(Refer Slide Time: 40:15)



Maximize $X_1 + X_2$

$7X_1 - 5X_2 \leq 7$

$-12X_1 + 15X_2 \leq 7$

$X_j \geq 0$ integer

$$\begin{array}{r|rrr}
 & & -X_1 & -X_2 \downarrow \\
\hline
 & & -1 & -1 \\
X_3 & 7 & 7 & -5 \\
X_4 & 7 & -12 & 15 \\
\leftarrow S_1 & 0 & -1 & 1
\end{array}$$

Now, let us spend few minutes on the all integer primal algorithm.

(No audio from 40:19 to 41:03)

So, let us take the same example that we saw in the previous lecture to understand the all integer primal algorithm. Now, we set up the initial table less than or equal to constraints, so you will have two slack variables X 3 and X 4. So we start with X 3 and X 4 here, and the decision variables will become minus X 1 minus X 2, it is a primal algorithm, primal - feasible, dual – infeasible. So we get a minus 1 minus 1 this will become 7, 7, minus 5, and 7, minus 12 and 15. Now, we once again have to find out, first we find out the entering variable in a primal algorithm. So, we would enter - these are already negative, so the smallest is the one that will enter. If we use this is like using z j minus c j for a maximization problem, so smallest z j minus c j will enter; if we use c j minus z j for a maximization problem, the largest c j minus z j will enter. In this case both are equal, so we could enter either this or this, just to show some illustration, let me enter X 2.

In a normal simplex we had only one possible, because this is negative, so you do not use this to compute the minimum ratio. So 7 by 15 will be the minimum ratio, and we would proceed; only problem is we would have fractions. So, since we want an all integer algorithm, once again by the same logic, a pivot of plus 1 is extremely suitable for an all integer primal algorithm, just as a pivot of minus 1 is suitable for an all integer dual algorithm. So to get a pivot of plus 1, what we do now is, we now identify this as the original pivot element, the easiest way to get a pivot of plus 1 is by simply divide it by 15 or use h is equal to 1 by 15 into the other one, and get it. So once again by doing this, this is an integer, this is the positive integer, this will not be 0, this will definitely be a positive integer, because it is a pivot element.

So 1 by this, is always a positive fraction between 0 and 1. A only case that can happen is, this can itself be 1, where h will be equal to 1; in this case, h is equal to 1 by 15, because this is 15. If this itself is 1, then h is equal to 1, you do not have to need a cut you can simply do on iteration. And only when this value is not 1, but another positive integer, then h will still be 1 by that positive integer, which is greater than 1; therefore, h will become a fraction between 0 and 1, strictly between 0 and 1. So, in this case, h is equal to 1 by 15. So use h is equal to 1 by 15, and into this pivot or source row to get 7 by 15 is 0 minus 1 plus 1; this is your slack variable S 1.

Now actually, this is the entering variable, this is the leaving variable, this is your pivot. So once again when we have a plus 1 pivot, what will happen is this feasibility will be maintained, because you are dividing by the pivot; this feasibility will be brought in, because you are dividing by negative of the pivot. So, minus 1 divided by minus 1, you give you plus 1, plus the all integer nature of this table is maintained. And in this algorithm, we do not do so much by trying to get the maximum like the largest increase rule, because you are anyway using the largest coefficient.

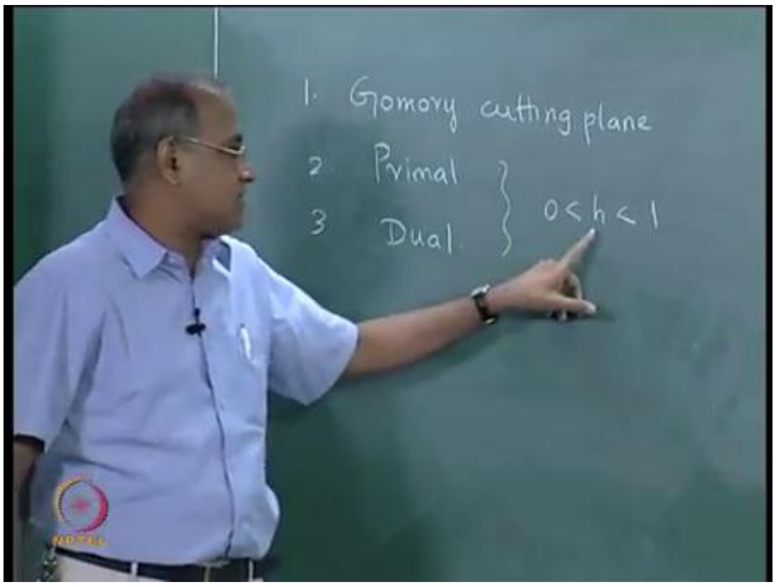
In the dual algorithm we started with first a leaving variable, and then an entering variable; so once the leaving variable is chosen in the dual algorithm, the entering variable is chosen in such a manner, there it gives largest increase. In the primal algorithm, the very choice first the entering variable will come in, and that entering variable is chosen based on the largest of the numbers, largest assuming c j minus z j in this case, these are negative numbers; so they represent z j minus c j therefore, we take the smallest of them.

Therefore, here in this case, both are equal, so just to illustrate I took this; so this is chosen based on what is called the largest coefficient rule; largest coefficient rule once again, if you treat them as c j minus z j, and values as plus 1 and plus 1, because they are z j minus c j, they are negative numbers; so you take the smallest of them and bring them inside. So already there is some kind of a local way to try and get a good variable. So that is already taken care of. So you do not worry so much about trying to get that largest increase, because you are working on largest coefficient. So all that is needed in this case is that the pivot element is a plus 1; that will definitely be ensured, because we take this number and divide.

As I said this number cannot be 0, because it is a pivot here, with this number cannot be negative, it has to be a strictly positive number. And if it is a strictly positive number, there are only two cases it can be 1, it can be greater than 1. So greater than 1 will give you h is equal to 1 by this number; if it is equal to 1, do not create a cut, simply do an iteration; then proceed. Very similar manner in the dual algorithm also, if it turns out that h is minus 1, h itself is minus 1; then you do not generate a cut; you simply do a dual simplex iteration and then proceed after that; because the integer character of the table is anyway going to be maintained, even without the addition of a cut; cut essentially, implies you are adding one more constraint, which means you are adding one more basic variable, you do not want to do that, if your h is 1 itself. So, that is another thing we learnt to do this.

So, before we go to the mixed integer programming, which is the next thing that we will do; let us have a very quick recap of what we have done in the last two lectures. So we started off by looking at three algorithms that essentially use the cutting plane idea, which means to an existing simplex table, you add an another constraint or a cut solve a resultant LP, such that at the end of it the LP solution is feasible both primal and dual feasible or it is feasible and satisfies the optimality condition; and gives an integer corner point, then it is optimum to the integer programming.

(Refer Slide Time: 48:32)



So, we saw 3 algorithms: the Gomory cutting plane algorithm, the all integer primal algorithm, and the all integer dual algorithm. Now, these two algorithms ensure that the integer nature of the table is maintained; whereas, the Gomory cutting plane algorithm allows fractional numbers to come into the table. We also saw the underlying relationship amongst the 3 by saying that whether it is a Gomory cut or whether it is a all integer dual cut or an all integer primal cut. We will have or satisfy this equation which is called the fundamental cut, which will ensure that, no integer point is eliminated by the addition of the cut. And we also saw that from an LP solution, if we directly take a source row which has a fractional value, and it should be having an integer value at the optimum.

And we take a source row, and then use that source row directly here, and put h is equal to 1, we get the Gomory cut. The all integer primal, and all integer dual cuts have h between 0 and 1 which are fractions, and exploits the condition that the lower integer value of h is 0. So, the lower integer value of h is 0, your primal cuts and dual cuts are centered around the only this term, and this term.

Now, depending on the primal feasibility, and the dual feasibility that is required for these algorithms; h values are suitably computed, primal and dual cuts are generated. And we can iterate, until we get to the optimum or not explicitly shown the iterations till we get to the optimum. The reader can continue to work out, because a mechanics of doing the all integer primal, and dual algorithms have already been described in earlier lectures. So, in the next lecture we will address mixed integer linear programming in detail.

.