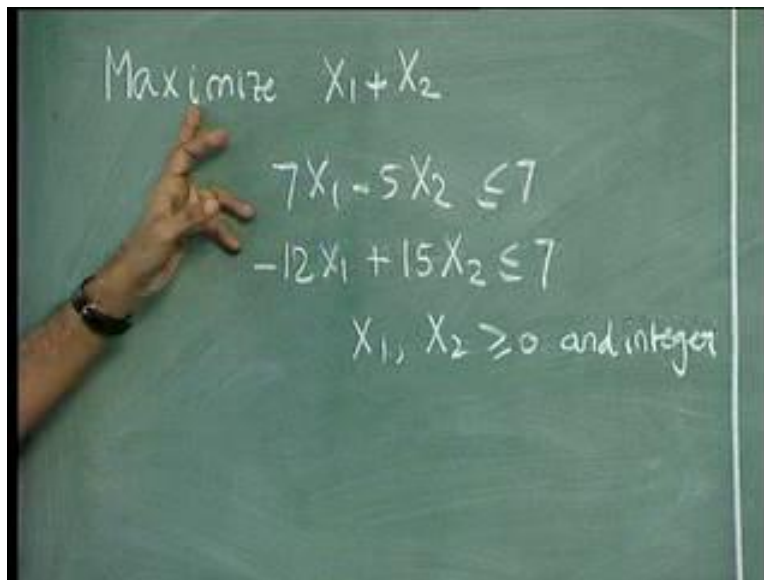


Advanced Operations Research
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

Lecture- 18
All-Integer Dual Algorithm

We continue the discussion on the all integer primal algorithm with this example which we have been working out.

(Refer Slide Time: 00:20)



The image shows a hand pointing to a chalkboard with the following text written on it:

$$\begin{aligned} &\text{Maximize } X_1 + X_2 \\ &7X_1 - 5X_2 \leq 7 \\ &-12X_1 + 15X_2 \leq 7 \\ &X_1, X_2 \geq 0 \text{ and integer} \end{aligned}$$

Two-variable problem with both constraints less than or equal to type, maximization with non-negative coefficients. We added slack variables X_3 and X_4 and started with X_3 and X_4 as the basic variables. We introduced many primal cuts and at some point, we had reached this iteration where the present solution is represented by X_1 equal to 1, Z equal to 1.

(Refer Slide Time: 00:42)

		$-S_5$	$-S_4$
X_0	1	5	-3
X_3	0	1	-3
$\leftarrow X_4$	19	-21	18
X_1	1	2	-1
X_2	0	3	-2
S_1	0	1	-1
S_2	0	1	-2
S_3	0	1	-3
$\leftarrow S_4$	1	-2	①

When we look at this iteration, we realise that here the dual is infeasible indicated by a negative sign. We enter this and then we have to find out the row from which the cut will be generated. In this entering column, there is only one element with a positive sign, rest of them are all negative. Therefore, this is the only row with which we can make a primal cut. This will become a temporary pivot. Dividing by 18 and introducing an S_6 , we would have 19 by 18 with a lower integer value of 1, minus 21 by 18 will have a lower integer value of minus 2 and 18 by 18 will give 1. This will be the pivot element and this is the leaving variable.

(Refer Slide Time: 1:40)

		$-S_5$	$-S_4$
X_0	1	5	-3
X_3	0	1	-3
$\leftarrow X_4$	19	-21	18
X_1	1	2	-1
X_2	0	3	-2
S_1	0	1	-1
S_2	0	1	-2
S_3	0	1	-3
$\leftarrow S_4$	1	-2	①

		$-S_5$	$-S_6$
X_0	4	-1	3
X_3	3	-5	3
$\leftarrow X_4$	1	① 15	-18
X_1	2	0	1
X_2	2	-1	2
S_1	1	-1	1
S_2	2	-3	2
S_3	3	-5	3
$\leftarrow S_4$	0	-2	① -1

We go back and continue the next table, we have a minus S_5 and a minus S_4 with X_0 , X_3 , X_4 , X_1 , X_2 , S_1 , S_2 and S_3 . S_4 has entered so we will have an S_4 and S_6 leaves, so S_6 will be here. The pivot element will remain as a pivot element with a 1. Rest of the elements in the pivot row are divided by the pivot. You get 1 and a minus 2. As far as this column is concerned, it is divided by the negative of the pivot; so you get a 3, 3, minus 18, 1, 2, 1, 2, and 3. This will become 1 minus minus 3 into 1 which is 4, 0 plus 3 is 3, 19 minus 18 is 1, 1 plus 1 is 2, 0 plus 2 is 2, 0 plus 1 is 1, 0 plus 2 is 2, 0 plus 3 is 3. We need to still evaluate this here. This will become 5 minus minus 3 into minus 2 is 5 minus 6, which is a minus 1. We get a minus 1 here, so 1 plus 6 is 7, minus 21 plus 36 is 15, 2 minus 2 is 0, 3 plus 4 is 7, 1 minus 2 is minus 1. 1 minus 4 is minus 3, 1 minus 6 is minus 5. 1 minus 6 is minus 5 and so on. X_2 is 3 minus 4 which is minus 1. We still have not reached the optimal because we find that this can enter. This enters and then we find that this (Refer Slide Time: 04:43) is the only one that can be a candidate. This will be a temporary pivot; this element will be a temporary pivot. We will introduce an S_7 here. There will be a 0, there is a 1 and there is a minus 2 that comes in. This is the actual entering variable and this will be the leaving variable, with this as the actual pivot.

We do one more iteration here.

(Refer Slide Time: 05:15)

	$-S_7$	$-S_6$
X_0	4	1
X_3	3	5
X_4	1	-15
X_1	2	0
X_2	2	1
S_1	1	1
S_2	2	3
S_3	3	5
S_4	1	2
S_5	0	1

X_0	1
X_3	0
X_4	19
X_1	1
X_2	0
S_1	0
S_2	0
S_3	0
S_4	1

We get minus S_7 , minus S_6 , X_0 , X_3 , X_4 , X_1 , X_2 , S_1 , S_2 , S_3 , S_4 and S_5 . Pivot element remains as a pivot element, rest of them are divided by the pivot. So, we get 0 and a

minus 2 here. This is divided by negative of the pivot, so 1, 5, minus 15, 0, 1, 1, 3, 5, and 2. Because of this 0, the same column will repeat here to get 4, 3, 1, 2, 2, 1, 2, 3, 1, and this will become 3, 3 minus minus 1 into minus 2, 3 minus minus 1 into minus 2 is plus 1. Therefore, we can stop here, because we have got the primal is feasible dual is also feasible. For the sake of completion, we would do 3 minus 10 is minus 7, minus 18 plus 30 is 12, 1, 2 minus 2 is 0, 1 minus 2 is minus 1, 2 minus 6 is minus 4, 3 minus 10 is minus 7 and 1 minus 4 is minus 3. Therefore, this is optimal; both primal and dual are feasible. Solution is X_1 equal to 2, X_2 equal to 2, and Z equal to 4, we are trying to maximise X_1 plus X_2 . This is how the all integer primal algorithm works.

Now, couple of issues; on one hand it is a very easy algorithm to implement, primal cut is also very easy to understand. The biggest hurdle in this algorithm is that, we can very quickly get into degeneracy which we have seen. For example, we have had some four iterations where the objective function did not move from 1 and after fourth or fifth iteration, from here it moved to four but then, again there is a degeneracy coming in. Every time a primal cut results in a 0 here, it will be degenerate and this will very quickly get into a degeneracy situation. You will realise that you are making too many unnecessary iterations before it converges to the optimum. This is the biggest drawback as far as this algorithm is concerned. You will end up having more iterations to perform. Nevertheless, if you want to work with an all integer algorithm or an all-integer cut, this algorithm is easy. Generating the all integer cut is also not very difficult; it is very intuitive and it is not very difficult. This is how the all integer algorithm works. The last part of the module on integer programming is to understand the all integer dual algorithm, which again we will explain using an example.

(Refer Slide Time: 09:32)

All integer dual algorithm

Minimize $8X_1 + 4X_2 + 6X_3$

$4X_1 + 5X_2 + 6X_3 \geq 18$

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

$4X_1 + 6X_2 + 3X_3 \geq 20$

$X_j \geq 0$ integer

We have an all integer dual algorithm and we explain it with a minimization problem. We minimize $8X_1$ plus $4X_2$ plus $6X_3$, subject to $4X_1$ plus $5X_2$ plus $6X_3$ greater than or equal to 18, $2X_1$ plus $3X_2$ plus $5X_3$ greater than or equal to 15, $4X_1$ plus $6X_2$ plus $3X_3$ greater than or equal to 20, X_j greater than or equal to 0 and integer. This is exactly the opposite problem. It is a minimization problem with all constraints greater than or equal to type. Typically, if it were a linear programming problem, we would have used a dual simplex to begin with. We develop what is called an all integer dual algorithm, which has more or less the same properties as dual simplex. It will start with a primal infeasible and dual feasible solution and proceed till you reach the optimum. We first create the table for this.

(Refer Time Slide: 11:06)

		$-X_1$	$-X_2$	$-X_3$	
z	X_0	0	8	4	6
≥ 18	X_4	-18	-4	-5	-6
≥ 15	X_5	-15	-2	-3	-5
≥ 20	X_6	-20	-4	-6	-3

We would start with X_4 , X_5 and X_6 as basic variables. We will have an X_0 here, X_4 , X_5 and X_6 as basic variables. Remember we would have minus X_1 here, minus X_2 and minus X_3 . When we write this as a maximization problem, you will have, maximize minus $8X_1$ minus $4X_2$ minus $6X_3$, would give us 8, 4, and 6 here. Remember it is a dual algorithm, so dual will be feasible, which is what you have got here. This will become minus 18, minus 4, minus 5, minus 6, minus 15, minus 2, minus 3, minus 5, and minus 20, minus 4, minus 6, and minus 3. When we write this in the standard form this is what we get for the given problem. The given problem is a minimization problem, with all constraints greater than or equal to type and the right hand side values are non-negative. This is what we have here. This will be at 0. The dual is feasible and the primal is infeasible. We have to generate a cut based on a dual simplex mode.

(Refer Slide Time: 12:55)

		$-X_1$	$-X_2$	$-X_3$	
X_0		0	8	4	6
X_4		-18	-4	-5	-6
≥ 18	X_5	-15	-2	-3	-5
≥ 15	$\leftarrow -X_6$	-20	-4	-6	-3
≥ 20					

We take that row which has the most negative element here as the row from which we generate the cut. You can either put an arrow here or just put a dash indicating that this row is the row from which the cut is going to be generated. The dual cut is slightly different from the primal cut. The all integer dual algorithm will have a pivot of minus 1. We have already seen that all integer algorithms will maintain the integer property by having a pivot as either plus 1 or a minus 1. Otherwise when you divide it with other numbers you will get fractions. So, initial table will be integer. In the all integer primal algorithm, the pivot element will be a plus 1, and in an all integer dual algorithm the pivot will be a minus 1. Pivot will be a minus 1 because in the next iteration, you will divide by the pivot. Some negative will become positive here and when you divide it by the negative of the pivot, the positive here will be retained. We need to maintain the feasibility of the dual. At the same time make one negative element positive in the next iteration.

What we do here is, the moment we decide this is the row based on which we are going to have a cut we look at all the negative elements in this row. In a subsequent iteration, you may have 0 or non-negative elements also; but we look at only the negative elements in this row. You have a 4, 6, and 3 with negative values coming in here. For all these negative values here, whatever be the negative value, go back and look at these. Now, you will look at all these three 8, 4, and 6, because the corresponding elements are all negative. Among these the smallest is 4. Remember

these will all be always positive or non-negative (Refer Slide Time: 14:54). The smallest is 4; divide everyone of this element by the smallest to get for example 2, 1, and 1, lower integer values.

(Refer Slide Time: 15:02)

		$-X_1$	$-X_2$	$-X_3$				
Row	X_0	0	8	4	6	2	1	1
	X_4	-18	-4	-5	-6	$1/2$	$1/6$	$1/3$
18	X_5	-15	-2	-3	-5			
15	$\leftarrow -X_6$	-20	-4	-6	-3			
20	S_1	-4	-1	-1	-1			

So 8 by 4 has a lower integer value of 2, 4 by 4 will have a lower integer value of 1, 6 by 4 will also have a lower integer value of 1. This becomes 2 by 4 which is half; leave out the negative sign 1 by 6 and 1 by 3. I repeat, having got this 2, 1, 1, now go back. Leave out the negatives. Take only those elements that have negative sign here and do this part so 2 by 4 is half, 1 by 6, and 1 by 3; the smallest is 1 by 6. If you take the smallest 1 by 6, now create a cut by multiplying this entire thing by the smallest number that you get, which is 1 by 6. In the subsequent iterations you will also understand it better. This will be an S_1 that comes in.

Minus 20 divided by 6 will be minus 4, minus 3 point something, which will give you a minus 4; minus 4 into 1 by 6 is minus 1, lower integer value again. This is a minus 1, this will be minus 1 again. You will have all minus 1 coming in. Remember this 2, 1, 1 was written based on the minimum here and in this process we will ensure that the pivot element is minus 1, corresponding to the minimum. It is that minimum that will be the entering one and this is the actual pivot that we have.

(Refer Slide Time: 17:05)

	$-X_1$	$-S_1$	$-X_3$
X_0	-16	4	4
X_4	2	1	-5
X_5	-3	1	-3
X_6	4	2	-6
X_2	4	1	-1

Continuing the iterations, we will have here minus X_1 , minus S_1 , minus X_3 . You have an X_0 here; you have an X_4 , X_5 , X_6 and X_2 . Pivot element becomes 1 by pivot, so it becomes minus 1 again. Rest of the elements in the pivot row is divided by the pivot to get plus 4, plus 1, plus 1. Here, divide by the negative of the pivot. Basically, you retain the same numbers 4, minus 5, minus 3, and minus 6. What we have done in the process is that, this minus 4 has become plus because the pivot is minus 1. You will see later that, the dual feasibility is maintained the way we carried out. The way we got these numbers 1 by 2, 1 by 6, and 1 by 3 dual feasibility will be maintained which we will see now. Now this will become... (18:18) 0 minus 4 into 4 will give you minus 16, minus 18 plus 20 is 2, minus 15 plus 12 is minus 3, minus 20 plus 24 is 4. You go back and check; X_1 equal to 2 or X_2 equal to 4 would give us Z equal to 16. So we are doing all right. Now, this one will be 8 minus 4 into 1 is 4, minus 4 plus 5 is 1, minus 2 plus 3 is 1, minus 4 plus 6 is 2, the dual feasibility is maintained.

Similarly, 6 minus 4 is 2, minus 6 plus 5 is minus 1, minus 5 plus 3 is minus 2, minus 3 plus 6 is plus 3. This is still primal infeasible because of this minus 3. This is the only row which has a negative and this is the only one which we will use to generate a cut. In this case, we had 3 negatives. The maximum negative element was used as the row from which the cut was written. Here, there is only one element and so we will use this row and generate a cut.

(Refer Slide Time: 20:16)

	$-X_1$	$-S_1$	$-X_3$			
X_0	-16	4	4	2		4 2
X_4	2	1	-5	-1		2 1
X_5	-3	1	-3	-2		2/3 1/2
X_6	4	2	-6	+3		
X_2	4	1	-1	1		
S_2	-2	0	-2	-1		

This has only two negatives which are minus 3 and minus 2. Corresponding elements are 4 and 2 here; divide by the smallest to get 2 and 1. Now, this becomes 2 by 3 and this becomes 1 by 2. 1 by 2 is smaller so 1 by 2 will be the one, which we will use. Now, multiply to create an S_2 ; minus 3 by 2 is minus 2, 1 into 1 by 2 is 1 by 2, which will become 0, minus 3 by 2 will become minus 2, minus 2 by 2 will become minus 2. I repeat again, this is the row with which we would generate a cut. There are two negatives, minus 3 and minus 2, corresponding to them, you have 4 and 2 here, which I have written down here; smaller one is 2. Divide these by the smaller to get 2 and 1. Go back to the corresponding negatives; you get 2 by 3 and 1 by 2. 1 by 2 is the smaller one, so multiply with 1 by 2; minus 3 into 1 by 2 is minus 3 by 2, lower integer value has minus 2. 1 into 1 by 2 is half, which will have a lower integer value of 0, minus 3 into 1 by 2 is minus 3 by 2 which becomes minus 2, and minus 2 into 1 by 2 will give us a minus 1. Pivot is minus 1, smallest will enter. This X_3 will enter and S_2 will leave. We will have another row that is created.

(Refer Slide Time: 22:14)

		$-X_1$	$-S_1$	$-S_2$	
Algorithm	X_0	-20	4	0	2
	X_3	4	1	-3	-1
$X_3 \geq 18$	X_5	1	1	1	-2
$X_3 \geq 15$	X_6	-2	2	-12	3
$X_3 \geq 20$	X_2	2	1	-3	1
Integer	X_3	2	0	2	-1

X_0 , X_4 , X_5 , X_6 , X_2 , and X_3 . I will have minus X_1 minus S_1 and minus S_2 . Pivot element will remain as pivot element minus 1. Pivot always becomes 1 by pivot, so minus 1 becomes minus 1. This is divided by the pivot, so I get 2, 0, and 2. This is divided by the negative of the pivot, so the same numbers come in 2, minus 1, minus 2, plus 3, plus 1. This becomes minus 16 minus 2 into 2 minus 4 is minus 20, 2 plus 2 is 4, minus 3 plus 4 is 1, 4 minus 6 is minus 2, 4 minus 2 is plus 2. I repeat; minus 16 minus 2 into 2 is minus 20, 2 minus 1 into 2 which is 2 plus 2 is 4, minus 3 plus 4 is 1, 4 minus 6 is minus 2, 4 minus 2 is 2. Because of this 0 the same column will repeat; 4, 1, 1, 2, 1.

This one will be 4 minus 2 into 2 is 0, minus 5 plus 2 is minus 3, minus 3 plus 4 is 1, minus 6 minus 6 is minus 12, minus 1 minus 2 is minus 3. This is what we get here. We have still not reached the optimum, because we have a negative element here. There is only one negative, therefore this is the row from which we will generate the all integer dual cut. This has only one element which is a minus 12, corresponding to that is actually 0. There is only one element corresponding to that is 0. The other thing that we need to do is, when we were here, for example, when we tried to generate the dual cut from this, we said there are 2 negatives; the corresponding were 4 and 2 and we divided with the smaller one so that we get a 1. If the smaller one turns out to be 0 then, you do not divide it by 0. Instead you keep it as 1 always. What we will effectively have in this case is the fraction that will be 1 by 12 and when we multiply, we will get a minus 1 that is coming in here. That is what will happen in this case.

When the smaller one becomes 0, actually you do not have anything. Whenever you have a situation, where the smaller one, or the smallest one happens to be a 0 that will become 1. That will be the basis on which we will generate the cut. What we will have here is a 1 by 12 that comes in, so we will have an S_3 , which is actually coming in.

(Refer Slide Time: 26:29)

	$-X_1$	$-S_1$	$-S_2$	
X_0	-20	4	0	2
X_4	4	1	-3	-1
X_5	1	1	1	-2
$-X_6$	-2	2	-12	3
X_2	2	1	-3	1
X_3	2	0	2	-1
$-S_3$	-1	0	-1	0

$1/12$

1 by 12; so minus 2 into 1 by 12, have a lower integer value of minus 1, 2 and 1 by 12 will give us a 0. 12 and 1 by 12 will give a minus 1, 3 by 12 will also be 0. Then we continue.

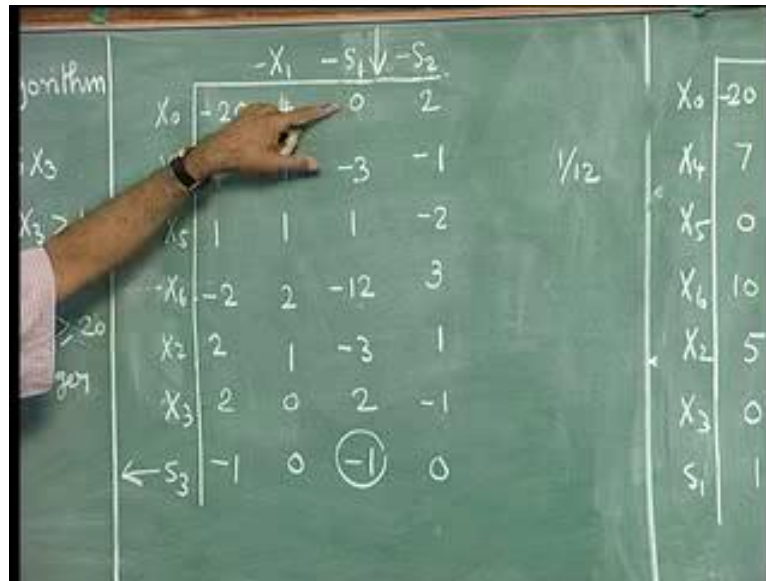
(Refer Slide Time: 28:06)

	$-X_1$	$-S_3$	$-S_2$	
X_0	-20	4	0	2
X_4	7	1	-3	-1
X_5	0	1	1	-2
X_6	10	2	-12	3
X_2	5	1	-3	1
X_3	0	0	2	-1
S_1	1	0	-1	0

We get a minus X_1 . This will be entering, this will be the pivot finally, this leaves; we get 20. We have $X_0, X_4, X_5, X_6, X_2, X_3$ and S_1 enters. I have S_3 and S_2 . Pivot becomes 1 by pivot, so I have a minus 1 here. Dividing by minus 1, I get 1, 0, and 0. Divide by negative of the pivot, I will get a 0 here, minus 3, 1, minus 12, minus 3, 2. Now, I write this 20 minus 0 into 1 is 20, 4 plus 3 is 7, 1 minus 1 is 0, minus 20, minus 2 plus 12, is 10. 2 plus 3 is 5, 2 minus 2 is 0. We can stop right here and say that we have reached the optimum, the reason being, we are performing a dual simplex iteration. Therefore, this will not violate the dual feasibility condition.

Nevertheless, we go back and check that, we have the same thing repeating. Because of this 0 and this 0, the same 2 columns will repeat. You will have a 4, 1, 1, 2, 1, 0, and a 2, minus 1, minus 2, 3, 1, minus 1 that comes in. Now, we have reached the optimum here. This is how the all integer dual algorithm works. This is how all integer dual algorithm works. The all integer dual algorithm has the same advantages and disadvantages as the all integer primal algorithm. In a way, the dual cut is little more involved than a primal cut. In the dual cut, we have to ensure that the pivot element is minus 1. We need to go back and do a couple of additional computations before we generate the dual cut. Primal cut was little more obvious than the dual cut.

(Refer Time Slide: 31:40)



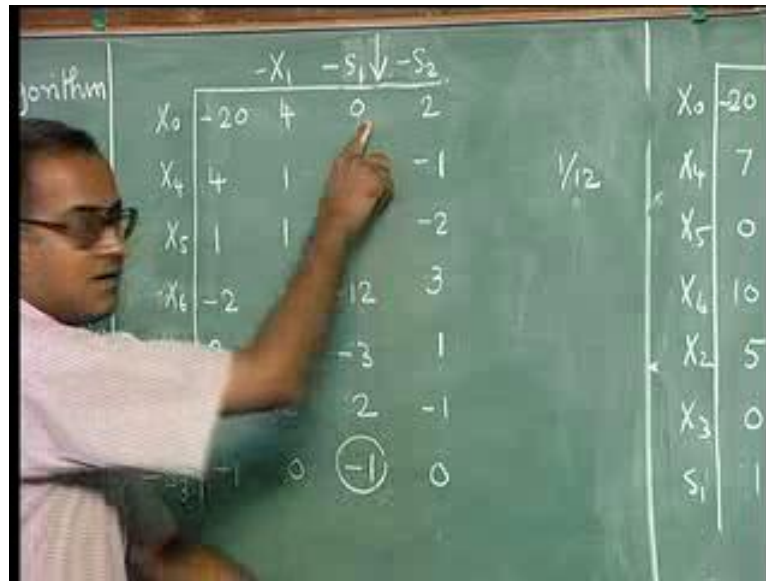
All these things that we did, we need to remember again that, this particular thing, that when the smallest element happens to be 0, we will have to keep it as 1 and it will automatically qualify to make the cut. When the smallest element is a non-zero element, then you divide so that you get a 1 and then you proceed. In every stage, you will only have to take the negative element and then generate the cut. The fraction that is obtained here is based on the negative elements that are there. Very intuitively you will understand that, the whole thing is centered around ensuring that the smallest among the negatives is the entering column. The pivot turns out to be minus 1 and the dual feasibility is maintained.

All these three things are done by the set of rules that we have used. We have not proved those rules but those will be the guiding rules based on which, we will generate a dual cut. In the next iteration we will realise that at least one negative element, because you are dividing with a minus 1, at least 1 negative element will become non-negative or positive. The dual feasibility will be maintained and there will be an increase or reduction, in this case, of the objective function. The all integer dual algorithm also suffers from the same problems of degeneracy that the all integer primal algorithm suffers. For example, we did have a degenerate iteration that comes in when this entering one has a 0, then you get into a degenerate situation.

The value of the objective function is not changed but this is non-optimal or infeasible to the primal, this is feasible to the primal. This is another example of degeneracy, but

degeneracy in a dual simplex algorithm. The primal degeneracy is slightly different from here. You have to remember that, it is again a case of degeneracy; degeneracy can happen. With some amount of experience or if you work out more problems, you will realize that, the all integer primal algorithm is more prone to degenerate iterations than the all integer dual algorithm, as a very general rule.

(Refer Slide Time: 34:04)

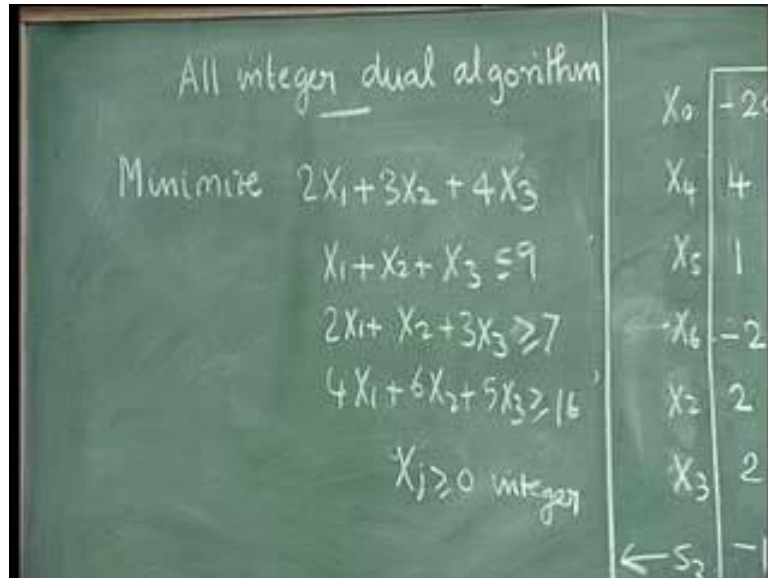


However, the moment you get an entering column with a 0, automatically you will get a degenerate iteration. Both these primal and dual algorithms have difficulties of quickly getting into degenerate solutions. Other than that, they provide a means by which you have all integer algorithms. You do not get into any fractions at any point in time. You carefully prepare or make the cuts, such that, the pivot element is a plus 1 for the primal algorithm and the pivot element is minus 1 for dual algorithm. The all-integer dual algorithm is ideally suited for a problem which has a minimization, which has all greater than or equal to constraints, non-negative right hand sides and minimization problem with non-negative coefficients in the objective function. Whereas, an all integer primal algorithm is suited to solve the dual of this problem, which is a maximisation problem with all constraints less than or equal to and a non-negative right hand side and all objective function coefficients non-negative.

What will happen if we have a mixed problem? Say we either have a maximization or a minimization; some of them turn out to be negative here, some of them turn out to be greater than equal to and some of them turn out to be less than or equal to. Can we

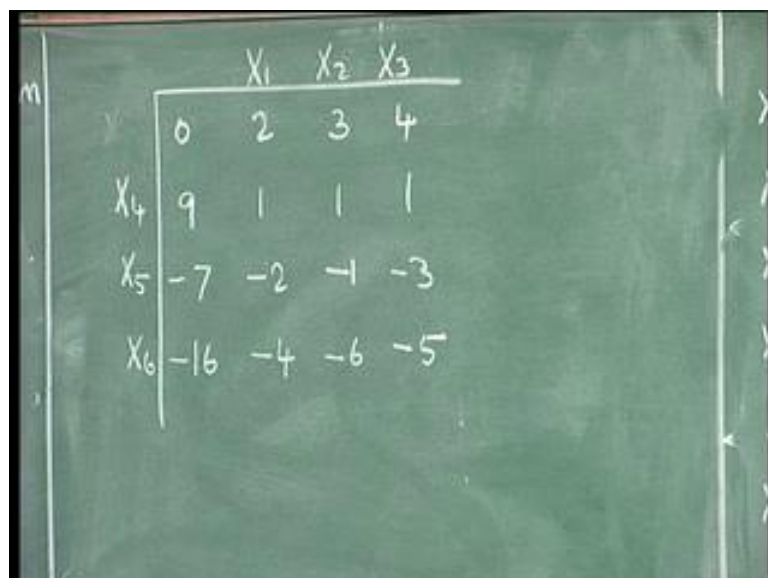
do or have an all integer algorithm which can handle that? We will just take an example; we will not go through the entire solution. We will just take an example and see how it looks like and then leave it at some point to solve. We will take an example here.

(Refer Slide Time: 36:00)



Suppose we have minimize $2X_1$ plus $3X_2$ plus $4X_3$; X_1 plus X_2 plus X_3 less than or equal to 9. $2X_1$ plus X_2 plus $3X_3$ greater than or equal to 7. $4X_1$ plus $6X_2$ plus $5X_3$ greater than or equal to 16. X_j greater than or equal to 0 and integer. Suppose we take a problem like this, we will define an X_4 , X_5 and X_6 , which will act as basic variables.

(Refer Slide Time: 37:00)



You will have an X_4 , X_5 and X_6 . It is a minimization. You will straight away have X_1 , X_2 , X_3 written here and you will have 2, 3, 4 and 0. Minimization implies dual feasible. You may have a negative also here, then this will become a minus 3. This will become, it is a less than or equal to constraint, so 9, 1, 1, 1. This is a greater than or equal to constraint minus 7, minus 2, minus 1, minus 3. It will become minus 16, minus 4, minus 6, minus 5. You can straight away proceed with an all integer dual algorithm. There are two negatives take them off negative and proceed. This would give us, all three are negative here, so, 2 3 and 4; this will become 1, 1, and 2, this will become 1 by 4, 1 by 6, 2 by 5. 1 by 6 will be the minimum and you can generate an all integer dual cut and you can proceed. If you get into a situation where some of these are negative, then you will have to redefine the whole thing carefully by checking out. For example, this being the smallest, among these things you will take only those which have positive and then you will proceed. You have to modify that rule suitably. It depends on whether you want to generate a dual cut or a primal cut. In this case the primal cut is not possible. Primal cut would mean that, you take an element whose dual is infeasible. Only the negatives will enter. A primal cut will first have an entering variable and then a leaving variable. A dual cut will first have a leaving variable and then an entering variable. In this case a primal cut is not possible because the dual is already feasible. In this case, only a dual cut is possible. If you had a minus 3 here, then a primal cut would have been possible.

(Refer Slide Time: 39:20)

	X_1	X_2	X_3	
X_4	0	2	-3	4
X_5	-7	-2	-1	-3
X_6	-16	-4	-6	-5

If you had a minus 3 here, then you can go back and do a primal cut. With this entering, there is only one candidate, so this will be the pivot and you will proceed; you can do that. You can also have a dual cut with this as the candidate, all 3 negative, but you will take only these two; 2 by 4, 2 by 5. 2 by 5 being smaller, you will generate a dual cut and you can proceed. Very similar to a linear programming problem with mixed type, you could have a situation where both primal and dual could be infeasible and so on, very similar to that situation, you may start with a primal cut or you may start with a dual cut.

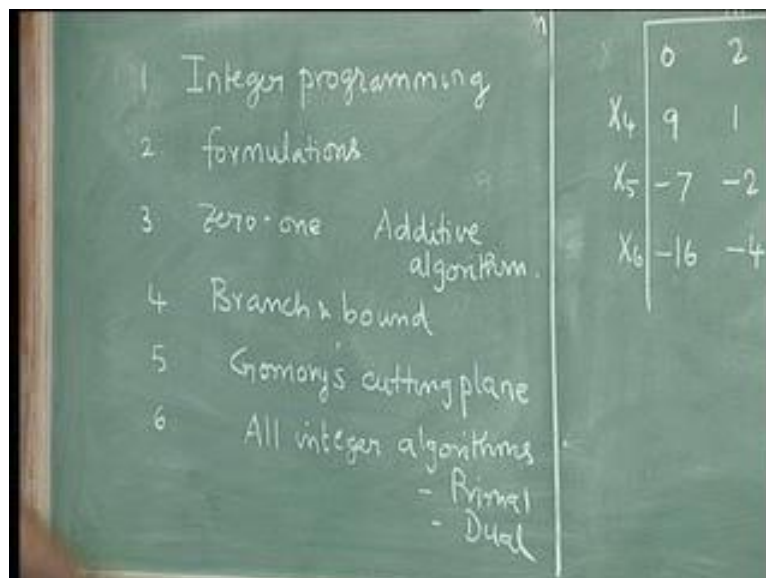
If the problem has an optimum then, you will realize that, at least one of the cuts will be possible. If both cuts are possible, as in this example, with a minus, then you could choose any one of them and proceed. But if you have a situation here (Refer Slide Time: 40:15) where a primal cut is not possible and only a dual cut is possible, you will have to proceed with the dual cut. If the problem has an optimum then, it will somehow take you to the optimum through a series of primal and dual cuts. As I said where both are possible, you can choose any one of them and proceed. The only thing that is not guaranteed is the number of iterations. It will eventually converge but, it might take longer or more number of iterations, depending on the problem situation.

The integers will at some point show infeasibility to the corresponding. After all in every stage you are only solving a linear programming; you are doing a simplex iteration. Therefore, as you as you add these cuts you will finally get into situation where, you do not seem to have a feasible point to the LP. May be a cut would make the whole thing infeasible and at that point infeasibility will be shown. Infeasibility will be shown again depending on what kind of a problem you are in. If you go back to infeasibility criterion of linear programming problems, there are two things: one is artificial variable is in the basis with a positive sign and we cannot have that happening here because, we are not dealing with the artificial variables at all. The only thing that would happen is, in both these, we are dealing with situations where you are introducing some cuts. In the dual simplex, infeasibility is always indicated by identifying a leaving variable and not identifying an entering variable; opposite of unboundedness, so something like that will happen here.

You will want to do a dual cut, you know that there is a variable here with a negative sign, but you are not able to get a corresponding variable to enter. That would mean

infeasibility and we have to think carefully about unboundedness. Unboundedness is the other way. You will be able to do a primal cut, but you enter something and you will not be able to find a leaving variable. That is how unboundedness and infeasibility will be detected in all these problems, whether it is an IP or whether it is an LP. The cut will take care in case of an IP. In case of a LP you do not introduce cuts, somewhere during the iteration, it will be taken care of. This is how the entire set of algorithms work. To quickly summarize what we have learnt in integer programming, we started with, reason to treat integer programming as a separate topic.

(Refer Slide Time: 42:55)



We said that, integer programming problems are those where the variables are restricted to be integers. We also saw the classification as all integer algorithms, mixed integer algorithms or mixed integer problems and 0, 1 problems. Then, we saw how, if you take integer programming as such as a problem and try to solve it by simplex or any LP algorithm, you will lose the convexity property of the feasible region. You will only solve those using tools of linear programming but then, do something in such a way that an LP optimum becomes integer optimum. When LP optimum becomes integer feasible it becomes integer optimum. That is a phenomenon that you observed in all of the Gomory cutting plane, as well as the all integer primal and all integer dual. You only introduced linear programming kind of constraints that involved continuous variables. At least when you introduced them into the simplex

table and proceeded, you always treated them as continuous; the way the thetas were defined, they were treated as continuous. We were going through a set of linear programming iterations but then when an LP optimum becomes integer feasible, it becomes integer optimum. So all those things we saw.

Then, we looked at some formulations and advantages of using integer programming as a way to model a set of situations. We saw examples where 0 1 was very powerful. You could model an either or situation, you could model k out of n constraints and so on. Then, we also saw how difficult optimization problems are modeled and formulated as integer programming problems, with the examples of the p median grouping problem, the travelling salesmen problem, line balancing problem, the scheduling problem and so on.

Then we moved to 0 1 problems and then we looked at the additive algorithms. It is a very powerful algorithm, simply because of the nature of the variables. Variables take only a 0 or a 1. Then we solved a standard problem, which was a minimization problem, with all constraints of the greater than or equal to type and all objective function coefficients non-negative. Then we showed that it is possible to have an efficient algorithm that uses only 3 or 4 vectors in terms of storage and memory. It was a great advantage to work. Then, we also saw how any given 0 1 problem can be converted to the standard form and how non-linear 0 1 problems can be converted to linear 0 1 problems and solved using the additive algorithm.

Then we saw the branch and bound algorithm, which can be used to solve all integer problems, as well as mixed integer problems. Branch and bound algorithm, essentially at every stage adds two constraints to the existing problem, both constraints are essentially bounds. We also saw that the constraints being bound essentially take off areas either as horizontal strips or as vertical strips in an existing feasible region and then solves the remaining problem. Since both the constraints are bounds one has to do one more iteration or more than one iterations. But one does not have the problem where the constraint size increases. You can straight away do an upper bounded simplex iteration. We have already seen how to do simplex algorithm with bounds. So one could do an upper bounded simplex iteration and keep continuing. The problem size does not increase at every node, even though the number of nodes can go up. We also briefly saw some issues on branching strategy bounding strategy and node

selection strategy along with this. We also said that this is a very efficient one because, we can handle mixed integer programming in the same way as it handles all integer problems plus the problem size does not increase at every iteration.

Then we looked at cutting plane algorithms. We saw the Gomory's cutting plane algorithm and in the Gomory's cutting plane algorithm, we solved the problem as the linear programming problem. Then if the LP optimum is infeasible to the IP, that is, if there are still variables which do not have integer values, we identify one of the variables, in general, the variables that has the largest fractional component and then generate a Gomory cut. A Gomory cut is different from a branch and bound cut. A Gomory cut is usually not a cut which is in the form of the bound. A Gomory cut is a typical cut which can involve more than one variable, whereas the branch and bound cut always involves only one variable. The Gomory cut would progressively remove areas from the feasible region and keep on chopping off areas from the feasible region, till the corresponding LP optimum is IP feasible and therefore we get to the optimum. The only drawback of the Gomory algorithm is, as the number of iterations increase, the size or the problem size at every iteration increases by one.

After looking at the cutting plane algorithm, we looked at all integer algorithms, all integer primal and all integer dual algorithms. The all integer primal algorithm is meant to solve a maximisation problem with less than or equal to constraints. An all integer dual algorithm is meant to solve a minimization problem with all greater than or equal to constraints. It is relatively easy to do the primal algorithm and generate a primal cut. The dual algorithm and the dual cut involved few more computations. Both the all integer primal as well as all integer dual algorithms have the problems of getting into degeneracy and the example that we illustrated the primal algorithm with ran into degeneracy right through the iteration and also at the optimum.

Similarly, the problem with which we demonstrated the dual algorithm also ran into degeneracy at the optimum. While these are efficient particularly for hand computation and particularly when we used this kind of representation of the simplex table where, as every time a cut is made, cut comes in the last or an additional row; the computations are easier. In some sense other than the advantage of hand computation, it is very difficult to say that, these could be better than a Gomory cutting plane algorithm or a branch and bound algorithm. When we do hand

computation to solve integer programming problems, the all integer algorithms are useful. When it comes to writing computer programs perhaps the Gomory's method or the branch and bound method would be better compared to all integer algorithms. With this we wind up the module on integer programming and then the next module that we look at are the network problems.