**Advanced Operations Research**

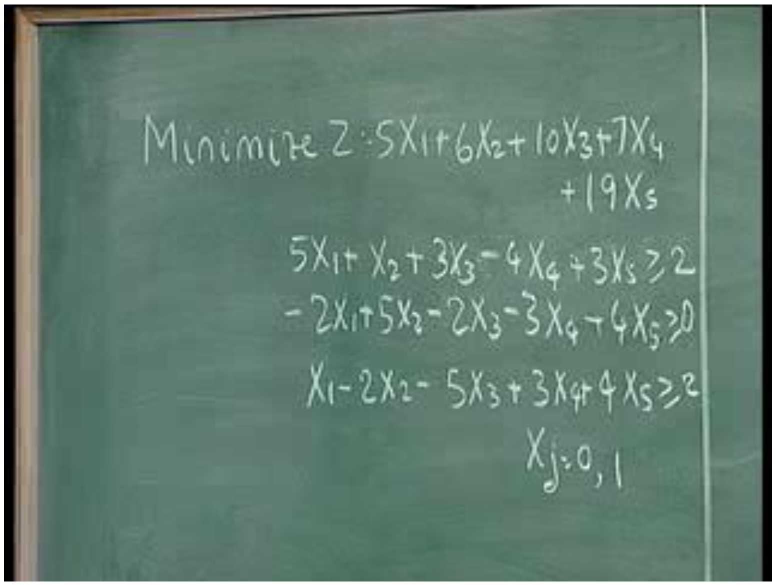
**Prof. G. Srinivasan**

**Dept of Management Studies**

**Indian Institute of Technology, Madras**

# Lecture - 14
# Solving Zero-One Problems

(Refer Slide Time: 00:14)



Minimize $Z = 5X_1 + 6X_2 + 10X_3 + 7X_4 + 19X_5$

$5X_1 + X_2 + 3X_3 - 4X_4 + 3X_5 \geq 2$

$-2X_1 + 5X_2 - 2X_3 - 3X_4 - 4X_5 \geq 0$

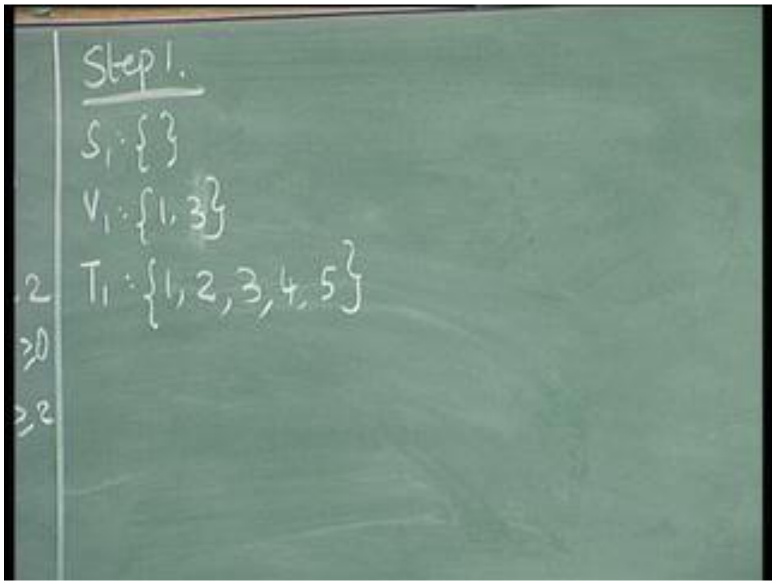
$X_1 - 2X_2 - 5X_3 + 3X_4 + 4X_5 \geq 2$

$X_j = 0, 1$

In the previous lecture, we were trying to solve this minimisation problem, which has all variables as 0, 1 variables. The standard problem has a minimisation objective, with objective function coefficients non-negative. All constraints are of the greater than or equal to type, no matter what the sign is on the left hand side or on the right hand side. If a solution with all variables equal to 0 is feasible, then it is optimal because of the nature of the objective function. Let us start the implicit enumeration algorithm again and try to solve this problem.

We start with something called step 1 or iteration 1.

(Refer Slide Time: 3:30)



Step 1.

$S_1 : \{\ \}$

$V_1 : \{1, 3\}$

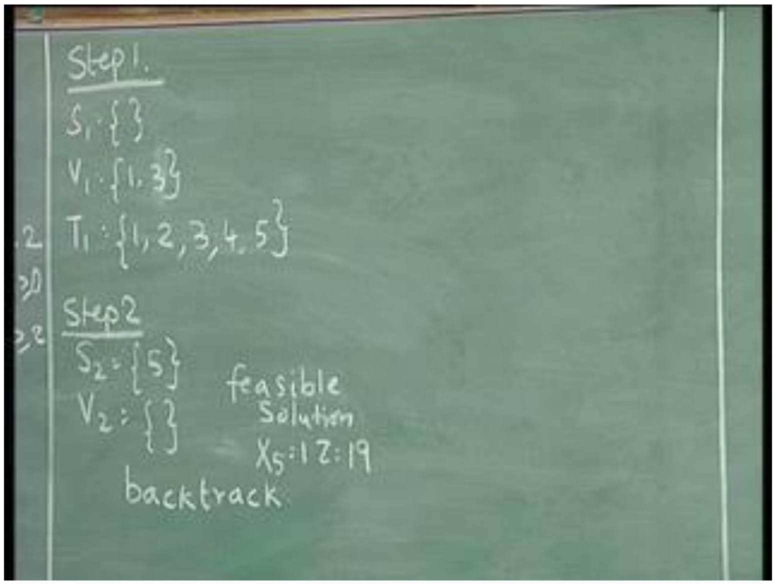
$T_1 : \{1, 2, 3, 4, 5\}$

We define, say $S_1$ as a null set, which means all variables are right now at 0. They have not been fixed to either 0 or 1. They are right now at 0. When all of them are 0, we have another set called a set of violated constraints. If all variables are at 0, constraints 1 and 3 are violated; we just write 1 and 3 here, it is a set comprising of the violated constraints. Then we want to get a feasible solution, so we define another set called a set of helpful variables. Any variable which has a strictly positive coefficient in any one of the violated constraints is helpful.

So, we have: $X_1$ is a helpful variable, so 1; $X_2$ is a helpful variable; $X_3$ is a helpful variable; $X_4$ is helpful because it has a plus 3 here and $X_5$ is also helpful because of both. Now we try to bring in feasibility. When it comes we will define it carefully. Right now none of the variables have been fixed to either 0 or 1. When we get into that situation, we will kind of redefine the set of helpful variables. We need to redefine it a little more precisely, which we will do when the right time comes. Among these helpful variables, we can take any one of them and try to fix it to a value 1. The important condition is a helpful variable should not already be fixed at 0 or 1; then it does not qualify. A variable that is so far not yet fixed and has is a helpful variable. But you will understand the first part once we start fixing variables. Among these, we could choose any one of them. Let us assume we start with $X_5$. This is not the only way of doing it, we could fix anyone of them to 1, but we would still choose variable $X_5$, because let us say, it has a positive coefficient in both and perhaps the sum is the biggest. For example, variable $X_1$ also has positive coefficients in both, but this has a larger sum. So let us assume that we fix 5 first.

Your step 2 will have $S_2$ equal to 5, indicating that variable $X_5$ is fixed to 1.

(Refer Slide Time: 5:13)



Step 1.

$S_1 : \{\ \}$

$V_1 : \{1, 3\}$

$T_1 : \{1, 2, 3, 4, 5\}$

Step 2

$S_2 : \{5\}$

$V_2 : \{\ \}$

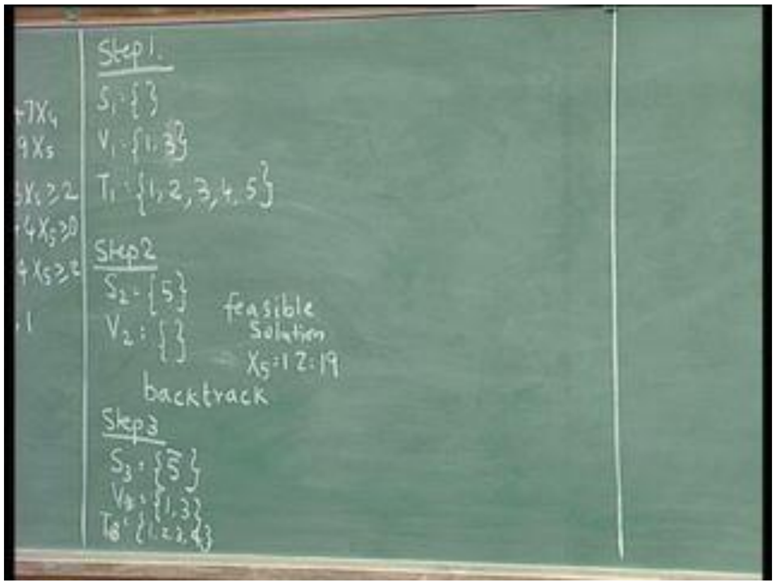
feasible Solution $X_5 : 1$ $Z : 19$

backtrack

The moment we fix variable $X_5$, and we have to find out $V_2$. $V_2$ is a set of violated constraints; this (Refer Slide Time: 03:47) is feasible; this is satisfied; this is also satisfied. So $V_2$ becomes a null set and the moment $V_2$ becomes a null set, which means no constraint is violated, you have a feasible solution. So you have a feasible solution and update that solution. So this solution is $X_5$ equal to 1, Z equal to 19. You update, initially you initialize, for example Z equal to infinity for a minimisation problem and because this 19 is less than the best value, you will update. Much later when you find another feasible solution, then you will verify the objective function value of that feasible solution with that of the existing list, and if it is better, you will update, otherwise you will leave it out. We update here with $X_5$ equal to 1 and Z equal to 19. There is no $T_2$, because $T_2$ finds a place only when there is something in $V_2$. So we have got a feasible solution. Now we need to check whether this feasible solution is optimal, or whether there exists any other feasible solution with a better value, in this case, a smaller value of the objective function. You do something called as back tracking.

Now you back track and when you backtrack, you create step 3.

(Refer Slide Time: 8:49)



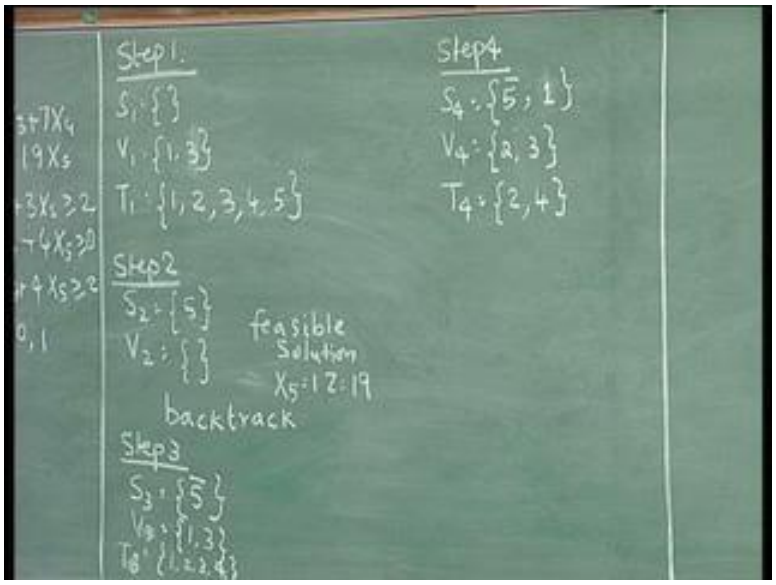
Go back to the S; create a $S_3$. Always come from the right side and look at the first variable that is there from the right side, which does not have a bar. That bar thing I will explain immediately. Right now 5 does not have a bar; I do not have a 5 bar, I just have a 5. So 5 is the only candidate on which I have to backtrack. When I backtrack, what I do is, I write this 5 as 5 bar. 5 bar implies that variable $X_5$ is now fixed at 0, so this is fixed. Now go back to $V_4$. This implies $X_5$ is fixed at 1, this implies $X_5$ is fixed at 0; Now go back and verify. $X_5$ is fixed at 0, the rest of the variables have not been fixed, but they are at 0. Therefore, we have a situation where all variables are at 0 with one of them, $X_5$ fixed, other variables are basically hanging and not fixed to anything but are at 0, which means once again your 1 and 3 will be violated. So 1 and 3 are violated because this is satisfied.

$T_3$ is a set of helpful variables. Now, 5 is not a candidate at all for a helpful variable because it has been fixed to a value. Now comes the correct definition of a helpful variable. A variable is helpful if it does not figure in the corresponding S, in either way, with a plus or a minus. Minus is the bar; plus implies it is fixed at 1; bar or minus implies it is fixed at 0. Those are the only two ways it can figure in S. So, all those variables that do not figure in S and have a positive coefficient in at least one of the violated constraints are helpful.

Now we will go back and write 1, 2, 3, 4, all of them qualify to be helpful variables. Whenever you backtrack, you eliminate a certain set of solutions. We will explain that part after we solve it. We will see what exactly happens; we will solve it by taking you through a branch and bound tree also. Now 1, 2, 3, 4 are there as helpful variables, you need to choose one of them and put them into the solution. This $X_1$ seems to be a very good candidate because it has a 5 plus 1. There is no hard and fast rule to choose one among this; you could choose anyone. You could choose them randomly, you could choose the one that has the best contribution and you could choose the first one that appears; you could do anything. Let us assume that we take variable $X_1$, because it has a large contribution of 5 plus 1.

We do that and when you write your $S_4$ you always have to write to the right of what you have which means this will become {5 bar, 4}.

(Refer Slide Time: 11:00)



$3+7X_4$
$19X_5$
$+3X_4 \geq 2$
$+4X_5 \geq 0$
$+4X_5 \geq 2$
$0, 1$

Step 1:
$S_1 : \{\}$
$V_1 : \{1, 3\}$
$T_1 : \{1, 2, 3, 4, 5\}$

Step 2
$S_2 : \{5\}$
$V_2 : \{\}$
feasible Solution
$X_5 = 1 \; Z = 19$
backtrack

Step 3
$S_3 : \{\bar{5}\}$
$V_3 : \{1, 3\}$
$T_3 : \{1, 2, 3, 4\}$

Step 4
$S_4 : \{\bar{5}, 1\}$
$V_4 : \{2, 3\}$
$T_4 : \{2, 4\}$

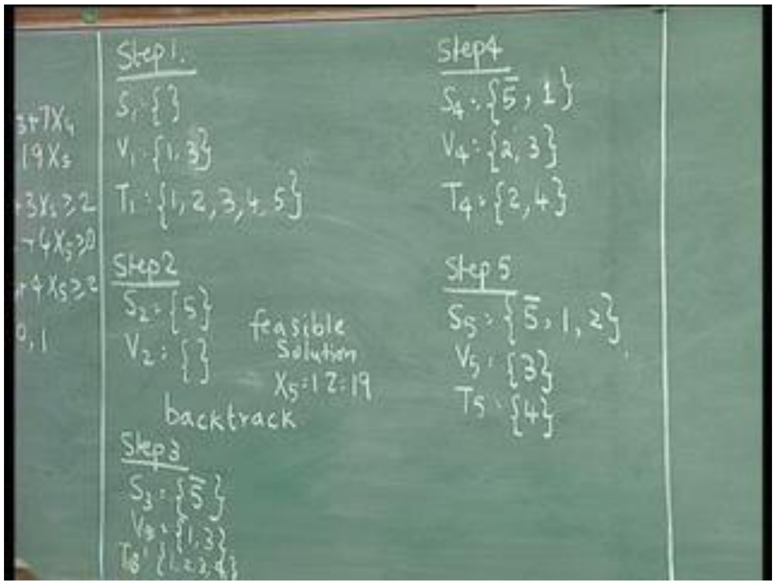
You will not write {4, 5 bar}, you will not write any of these and particularly the S, in an order other than this order. Whenever you add something into a solution, it always comes after the previous step solution. You should not do 4 and 5 bar, we will see why later. First let us complete this and then explain why and what we are doing at each stage. We decided to choose 1. So this will become 5 bar and 1 (Refer Slide Time: 9:39). Variable $X_1$ comes in, so it is 5 bar and 1. Now $X_5$ is fixed at 0, $X_1$ is fixed at 1; so this is satisfied, this is violated, this is satisfied. So $V_4$ is 2, because second constraint is violated by this solution. Third constraint is also violated; 1 greater than 2, so we will have 2,3, so both are violated by this. Go back and prepare a set of helpful variables. Both 5 and 1 cannot figure in your list of helpful variables, should only look at 2, 3 and 4. Now 2 is helpful here; 3 is not helpful because both are negative, 4 is again helpful. At this stage, you can go back and put one of the helpful variables and proceed further; you can do that.

You can try something else if you want; the algorithm allows you both. In its natural course, what you would do is, you will put one of the helpful variables here and proceed as you did, but you can try this also. For example, I have a situation where I have 2 and 4 as helpful variables; 5 is already fixed at 0; 1 is fixed at 1. Now, as far as this constraint is concerned, I have 2 and 4 as helpful variables, now it is still all right. If we put $X_2$ alone to 1, I may be able to get feasibility. I come back to this and let us see where I am. 5 is fixed at zero, so we need to proceed.

For example, what you can try and do here is, if you are able to show that if I take a violated constraint and as far as that constraint is concerned, even if we put all the helpful variables, I do not get feasibility, then you can use the earlier result of fathoming by infeasibility. You can try doing that at every iteration. Only thing is you lose some CPU time trying to verify. Again there is no hard and fast rule that says this is exactly the way it is. Particularly, when you write an algorithm, you have no way to see the actual numbers as they are, when you run it on a computer. So for a smaller problem, you can afford to do it; otherwise, you can have a standard routine which does this at every stage and takes you through. Sometimes it is advantageous, sometimes it is not. We will go through this 2 and 4. Now the violated constraints are this, so you can either put $X_2$ or put $X_4$ into the solution. Let us assume that you put $X_2$ into the solution.

So step 5 is, $S_5$ equal to {5 bar, 1, 2}.

(Refer Slide Time: 13:55)



$3+7X_4$

$19X_5$

$+3X_1 \geq 2$

$+4X_5 \geq 0$

$+4X_5 \geq 2$

0, 1

Step 1.

$S_1: \{\}$

$V_1: \{1, 3\}$

$T_1: \{1, 2, 3, 4, 5\}$

Step 2

$S_2: \{5\}$

$V_2: \{\}$

feasible Solution $X_5 = 1\ Z = 19$

backtrack

Step 3

$S_3: \{\bar{5}\}$

$V_3: \{1, 3\}$

$T_3: \{1, 2, 3, 4\}$

Step 4

$S_4: \{\bar{5}, 1\}$

$V_4: \{2, 3\}$

$T_4: \{2, 4\}$

Step 5

$S_5: \{\bar{5}, 1, 2\}$

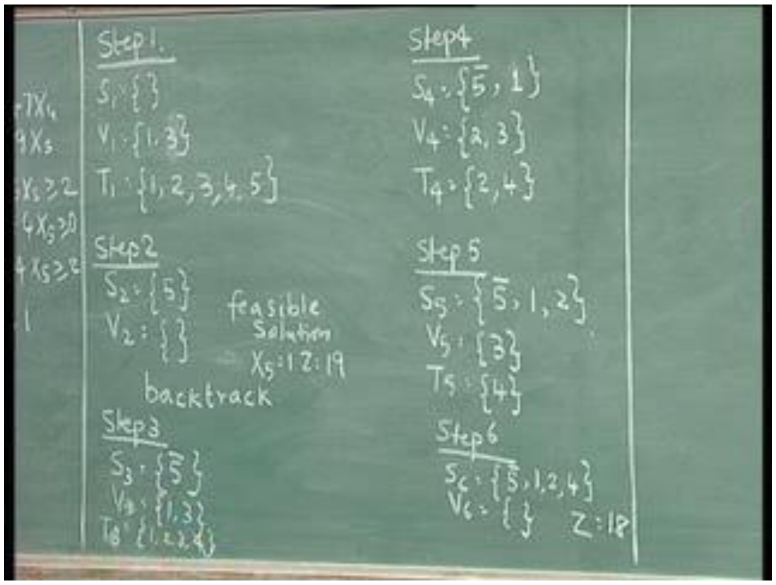
$V_5: \{3\}$

$T_5: \{4\}$

Now $V_5$ is: 1 and 2 are fixed at 1, this (Refer Slide Time: 13:23) is satisfied; 1 and 2 are fixed at 1, this is satisfied; 1 and 2 are fixed at 1, this is not satisfied. So I have $V_5$ equal to 3. Set of helpful variables $T_5$ for this is: already 1, 2 and 5 are in the solution; so, only 3 and 4 are available. 3 is not a helpful variable, 4 is; so, we will have 4 as a set of helpful variable. At least here you can check whether putting the helpful variable to 1 can bring feasibility. Right now it can, because 1 and 2 are fixed at 1, which is a minus 1. Putting this can bring in feasibility for this constraint. So go ahead and do it.

Step 6 will be: $S_6$ equal to {5 bar, 1, 2, 4}.

(Refer Slide Time: 14:54)



Step1.

$S_1 : \{\}$

$V_1 : \{1, 3\}$

$T_1 : \{1, 2, 3, 4, 5\}$

Step2

$S_2 : \{5\}$

$V_2 : \{\}$

feasible Solution $X_5 : 1$ $Z : 19$

backtrack

Step3

$S_3 : \{\bar{5}\}$

$V_3 : \{1, 3\}$

$T_3 : \{1, 2, 3, 4\}$

Step4

$S_4 : \{\bar{5}, 1\}$

$V_4 : \{2, 3\}$

$T_4 : \{2, 4\}$

Step 5

$S_5 : \{\bar{5}, 1, 2\}$

$V_5 : \{3\}$

$T_5 : \{4\}$

Step 6

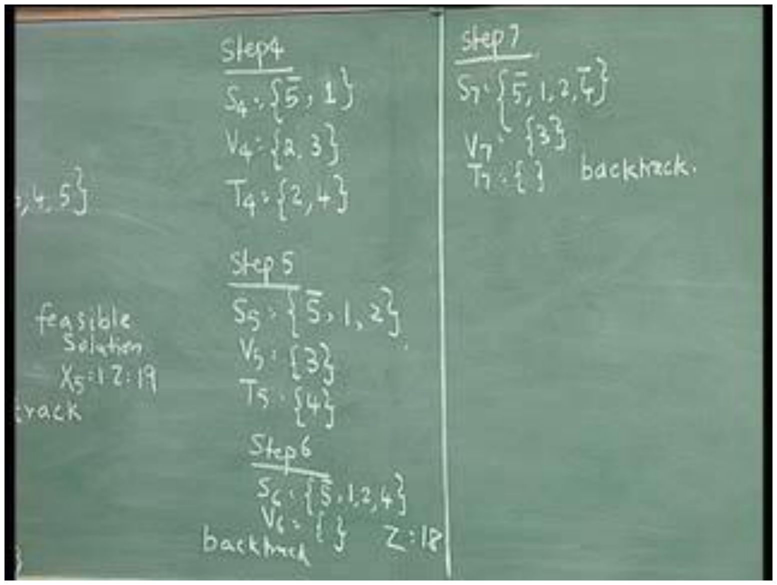
$S_6 : \{\bar{5}, 1, 2, 4\}$

$V_6 : \{\}$ $Z : 18$

$V_6$: now 1, 2, and 4, 6 minus 4 is 2, which is satisfied, 3 minus 3 is 0 is satisfied; 1 2 and 4 is satisfied, 1 2 and 4 is also satisfied. So I have a feasible solution here at this (Refer Slide Time: 14:56). So I try to find out the value of the objective function for this. This gives me: 1, 2, and 4 is 5 plus 6 equal to 11 plus 7, 18; so I have a feasible solution with Z equal to 18, which is better than the solution with Z equal to 19. So I update a solution with Z equal to 18, which implies $X_1$ equal to $X_2$ equal to $X_4$ equal to 1 and Z equal to 18. The moment I have a feasible solution, I backtrack again. So I backtrack here.

Step 7: $S_7$ will be, now go back, come from the right and the first variable which does not have a bar is the one to be backtracked.

(Refer Slide Time: 17:12)



$,4,5\}$

feasible Solution $X_5 : 1$ $Z : 19$

track

Step 4

$S_4 : \{\bar{5}, 1\}$

$V_4 : \{2, 3\}$

$T_4 : \{2, 4\}$

Step 5

$S_5 : \{\bar{5}, 1, 2\}$

$V_5 : \{3\}$

$T_5 : \{4\}$

Step 6

$S_6 : \{\bar{5}, 1, 2, 4\}$

$V_6 : \{\}$

backtrack $Z : 18$

step 7

$S_7 : \{\bar{5}, 1, 2, \bar{4}\}$

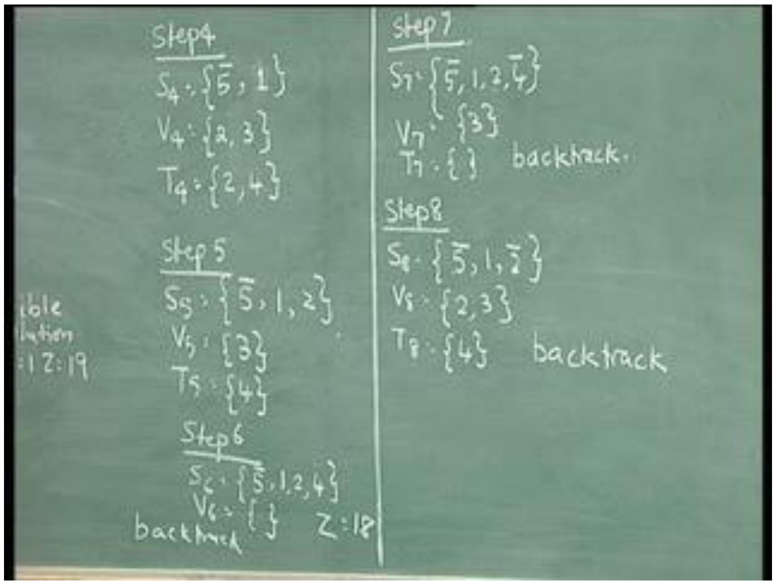
$V_7 : \{3\}$

$T_7 : \{\}$ backtrack.

As you come from the right, this is the variable to be backtracked. So I have a solution: {5 bar, 1, 2, 4 bar}. If you look at it very carefully, your $V_7$ will be this, because this also implies 5 bar, 1, 2; 4 bar implies 4 at 0. So, $V_5$ will be the same as $V_7$, you can either substitute or straight away take it from here, so $V_7$ will be 3. Now go back to this 3, there is no helpful variable. Even though there is a variable 3, there is no helpful variable, so $T_7$ becomes null set. When the set of helpful variables becomes a null set, you again backtrack. So this is a second reason as to why you backtrack. One is when you have a feasible solution you backtrack; second is when you do not have a helpful variable, you backtrack.

Now this is an interesting thing about backtracking that is going to happen. Now you will go back and backtrack on 2. What you will do is, {5 bar, 1, 2 becomes 2 bar} and once you backtrack, you have to leave whatever is to the right of it. In this case, you will not write this 4 bar again. You leave whatever is to the right of it.

(Refer Slide Time: 20:27)



ible
lution
=12:19

Step4

$S_4 = \{\bar{5}, 1\}$

$V_4 = \{2, 3\}$

$T_4 = \{2, 4\}$

Step 5

$S_5 = \{\bar{5}, 1, 2\}$

$V_5 = \{3\}$

$T_5 = \{4\}$

Step6

$S_6 = \{\bar{5}, 1, 2, 4\}$

$V_6 = \{\ \}$ $Z = 18$

backtrack

Step 7

$S_7 = \{\bar{5}, 1, 2, \bar{4}\}$

$V_7 = \{3\}$

$T_7 = \{\ \}$ backtrack.

Step8

$S_8 = \{\bar{5}, 1, \bar{2}\}$

$V_8 = \{2, 3\}$

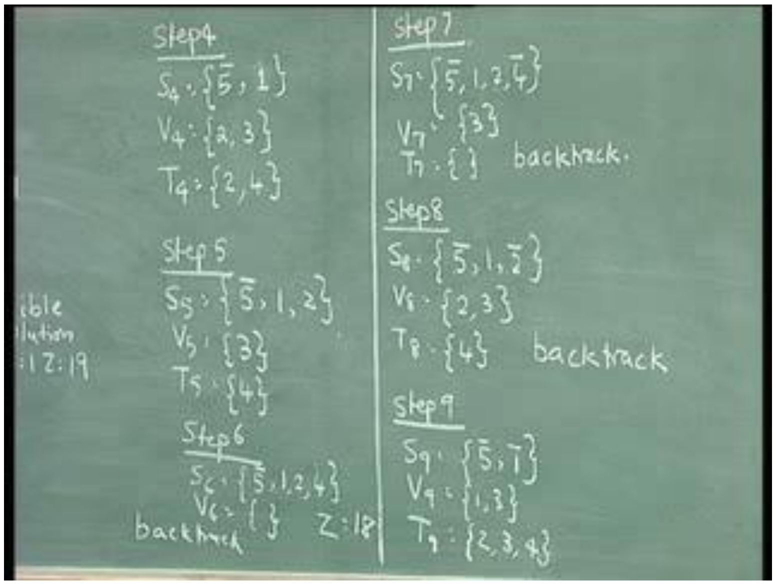
$T_8 = \{4\}$ backtrack

Go back, you have consistently done this; there is only one, so there is nothing to the right of it. Similarly, there is nothing to the right of it. This is the first time you find something to the right of it, so leave out that 4. We will see each one of this in detail, as to why we are doing it after we finish solving the problem. $V_8$ will be: this is like fixing 1. This is like your 5 bar 1, because this is basically 2 at 0. So this is 5 bar and 1, so you will have 2 and 3 as violated constraints. You can go back and check. This is satisfied, this is violated, and this is violated. We now go back to our set of helpful variables: $T_8$ will be, already 5, 1, and 2 are fixed, only 3 and 4 are remaining; both are not helpful here with respect to this. As far as this is concerned both are not; as far as this concerned 4. Now, go back and apply the other results. There is a helpful variable. Putting this helpful variable to 1 under the present condition cannot bring feasibility here or as far as this violated constraint is concerned, there is no helpful variable, which mean the same. This is exactly what we had. This is a situation where there is a violated constraint and there is no helpful variable for that violated constraint. That is one.

But there is a helpful variable 4, putting it to 1 can bring feasibility, because I have 1 plus 3; but when we put this to 1, this is definitely going to be violated. So I can either put this, have 1 more iteration and then go back, or I can backtrack straight away here, because I have a violated constraint. I do not have a helpful variable for this violated constraint, therefore I can backtrack. There is no point in proceeding. So I backtrack here under the assumption that I have a violated constraint - a constraint that is violated by the present solution and there is no helpful variable, as far as that constraint is concerned, I can backtrack.

Now I go back and backtrack, I do step 9.

(Refer Slide Time: 21:29)



Step4

$S_4 : \{\bar{5}, 1\}$

$V_4 : \{2, 3\}$

$T_4 : \{2, 4\}$

Step 5

$S_5 : \{\bar{5}, 1, 2\}$

$V_5 : \{3\}$

$T_5 : \{4\}$

Step6

$S_6 : \{\bar{5}, 1, 2, 4\}$

$V_6 : \{\}$ $Z = 18$

backtrack

ible

lution

12:19

step 7

$S_7 : \{\bar{5}, 1, 2, \bar{4}\}$

$V_7 : \{3\}$

$T_7 : \{\}$ backtrack.

Step8

$S_8 : \{\bar{5}, 1, \bar{2}\}$

$V_8 : \{2, 3\}$

$T_8 : \{4\}$ backtrack

Step9

$S_9 : \{\bar{5}, \bar{1}\}$

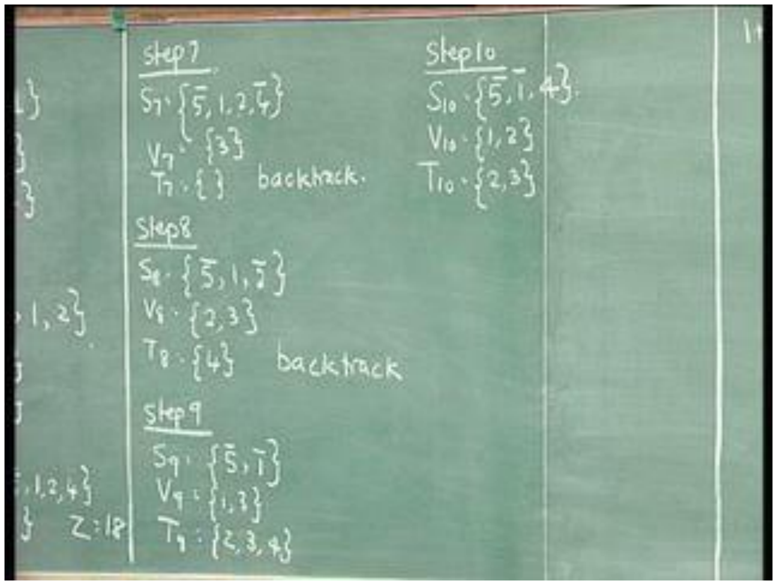
$V_9 : \{1, 3\}$

$T_9 : \{2, 3, 4\}$

I come from the right, so $S_9$ will be {5 bar, 1 bar}. I will leave the 2 behind. So I will have a 5 bar and 1 bar. So your 5 bar and 1 bar is like your null set straight away. I have 1 and 3 violated. $T_9$, set of helpful variables will be, I should only look at 2, 3, and 4. 5 bar, 1 bar implies null set; all of them are at 0. So 1 and 3 will be violated. Now I have to look at 2, 3, and 4; so 2 is helpful, 3 is helpful, and 4 is helpful. I do 2, 3, and 4. Now go back to this. I already have fixed these two at 0. I have 2, 3 and 4, which are here; this is violated. I can still try and put $X_4$ and get feasibility here. It might have other indication but let us not worry about that. In principle, there is a helpful variable; by putting it to 1, I can get feasibility here, so I try going to 4.

I have step 10, $S_{10}$.

(Refer Slide Time: 23:06)



Step 7

$S_7 \cdot \{\bar{5}, 1, 2, \bar{4}\}$

$V_7 \cdot \{3\}$

$T_7 \cdot \{\ \}$ backtrack.

Step 8

$S_8 \cdot \{\bar{5}, 1, \bar{3}\}$

$V_8 \cdot \{2, 3\}$

$T_8 \cdot \{4\}$ backtrack

Step 9

$S_9 \cdot \{\bar{5}, \bar{1}\}$

$V_9 : \{1, 3\}$

$T_9 : \{2, 3, 4\}$

Step 10

$S_{10} \cdot \{\bar{5}, \bar{1}, 4\}$

$V_{10} \cdot \{1, 2\}$

$T_{10} \cdot \{2, 3\}$

$Z : 18$

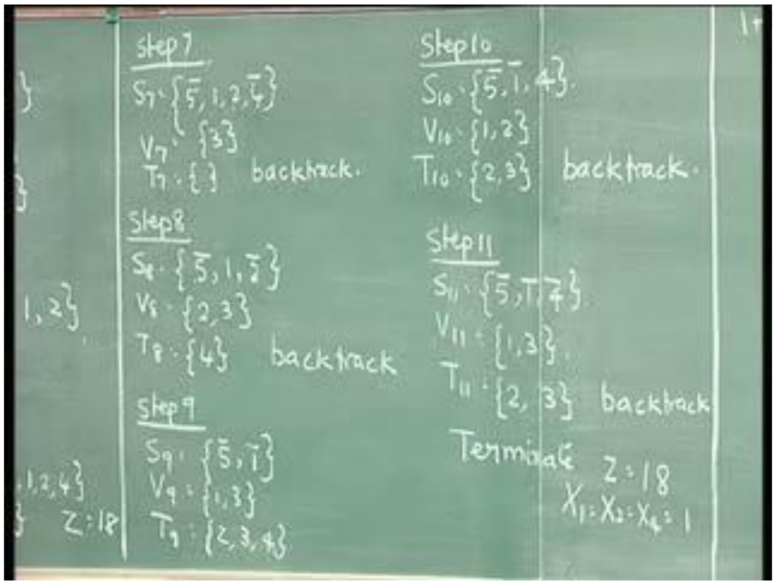
I can put variable $X_4$ here. Out of 2, 3, and 4, 2 has a minus 1 here, 3 has a minus 2 here, 4 also has a minus 1 here, maybe I can try 4. So $S_{10}$ will be {5 bar, 1 bar, 4}. $V_{10}$, this is like fixing variable $X_4$ to 1; this is violated, this is violated, this is satisfied, so 1 and 2 are violated. $T_{10}$, I can only think in terms of 2 and 3, because 1, 4, and 5 are already fixed. So I look at 2 and 3; 2 is helpful, 3 is also helpful. Now 2 is helpful, so I can have 2 and 3 here; both are helpful. Let us assume that I will try to put variable $X_2$ here. I have 5 bar, 1 bar and 4. Step 9 itself, I can backtrack. This is a small problem. This is something which we have to be careful about. Let us go back to this reasoning.

Let us go to step 9 and see where we are. 5 bar and 1 bar; so violated constraints are 1 and 3, helpful variables are 2, 3, and 4. The logic is, when I say 2, 3, and 4, I brought in 4, because I look only at this constraint in isolation and believe that I can bring feasibility by fixing this $X_4$ to 1. When I do that, I do not go back and check at the other violated constraint, and check that, when I put $X_4$ to 1, then by putting the rest of them, I cannot get that. That argument you normally do not do here. It is not that the argument is incorrect; it is just that, if you want to write it as an algorithm, for a problem which has more variables and more constraints, at every iteration you end up spending a whole lot of time checking this.

The only thing you have to do is you have to it very judiciously and common sense will tell you, that at any point in time look at only one constraint. Do not see the impact of putting a helpful variable through a constraint on some other one. Normally we do not do that. That is the reason why we entered this 4. We could have backtracked. If you want to see the effect of putting $X_4$ on this, then you could backtrack but we normally do not do that. Here, you can backtrack because the two helpful variables 2 and 3. I go to the first constraint, already I have $X_4$ fixed at 1; so by putting both the helpful variables I cannot get feasibility; therefore I backtrack. Please go back one at a time; always constraints are handled one at a time. Go back; by putting both the helpful variables, I cannot get feasibility because I want this 2 here, therefore, I backtrack here.

So my step 11 will have: $S_{11}$ equal to {5 bar, 1 bar, 4 bar}, which is again like a null set, because all of them are at 0.

(Refer Slide Time: 27:59)



step 7

$S_7 \cdot \{\bar{5}, 1, 2, \bar{4}\}$

$V_7 \cdot \{3\}$

$T_7 \cdot \{\ \}$ backtrack.

step 8

$S_8 \cdot \{\bar{5}, 1, \bar{2}\}$

$V_8 \cdot \{2, 3\}$

$T_8 \cdot \{4\}$ backtrack

step 9

$S_9 \cdot \{\bar{5}, \bar{1}\}$

$V_9 \cdot \{1, 3\}$

$T_9 \cdot \{2, 3, 4\}$

Step 10

$S_{10} \cdot \{\bar{5}, \bar{1}, 4\}$

$V_{10} \cdot \{1, 2\}$

$T_{10} \cdot \{2, 3\}$ backtrack.

Step 11

$S_{11} \cdot \{\bar{5}, \bar{1}, \bar{4}\}$

$V_{11} \cdot \{1, 3\}$

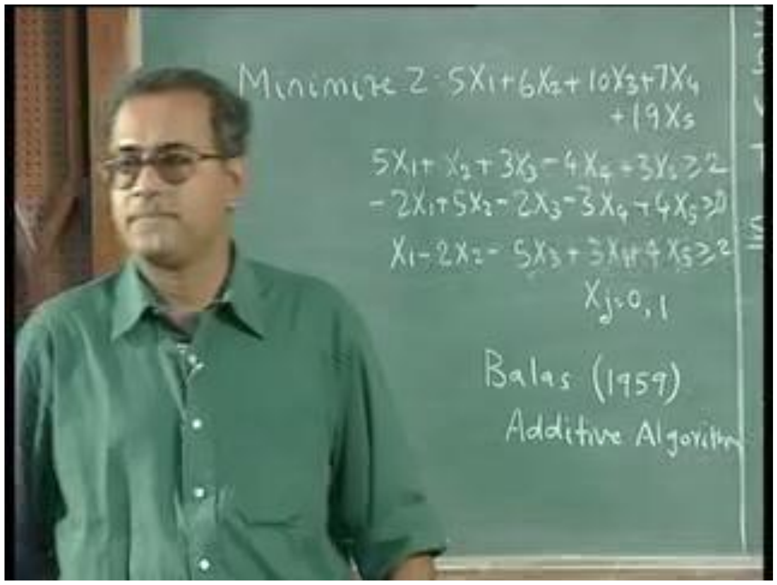
$T_{11} \cdot \{2, 3\}$ backtrack

Terminate $Z = 18$

$X_1 = X_2 = X_4 = 1$

So 1 and 3 will be violated. Now go back to $T_{11}$. I have only 2 and 3 available as helpful variables in the list. Both 2 and 3 qualify because they have a plus coefficient here. So 2 and 3 are helpful. But you realise that as far as the third constraint is concerned, it is violated and it does not have any helpful variable, I can backtrack. So I go back and backtrack. Now I want to backtrack, but I cannot backtrack because I do not have a variable on which I can backtrack. Only when you have at least one variable which is fixed at 1, I backtrack. Now all of them here are fixed at 0, so I cannot backtrack. This algorithm will terminate when no backtracking is possible. That is your termination condition. No backtracking is possible, the algorithm terminates. The best solution that you have got is the optimal solution. You have Z equal to 18 with $X_1$, $X_2$, and $X_4$ equal to 1. So, this is how this algorithm works.

(Refer Slide Time: 28:10)


Minimize Z = 5X1 + 6X2 + 10X3 + 7X4 + 19X5
5X1 + X2 + 3X3 − 4X4 + 3X5 ≥ 2
−2X1 + 5X2 − 2X3 − 3X4 + 4X5 ≥ 0
X1 − 2X2 − 5X3 + 3X4 + 4X5 ≥ 2
Xj = 0, 1
Balas (1959)
Additive Algorithm

This is a very old algorithm called Balas additive algorithm. The name of the person is Balas; it is not apostrophe. The name of the person is Balas. It is a 1959 algorithm. It is called an additive algorithm for zero-one. It is also called implicit enumeration algorithm for zero-one. The original algorithm did not, for example, provide any guidelines on when do you use this second fathoming strategy judiciously. There are many things that are being kind of kept hanging in loose. Many things are fixed and certain. Among them are the rules to backtrack. When do you backtrack is fixed, termination condition is well known and fathoming by feasibility is obvious. When we get a feasible solution, I fathom or I backtrack; we will come back to that. What is not explicitly stated is; number one, the thing that we tried to do in step 9. If you know the problem well enough or if you can see what is happening, as you see on the board, you know that, fixing $X_4$ equal to 1 is going to have a damaging effect on some other constraint. When you cannot see the problem when it is being solved or when you write a code to solve this, there is no way for you to know, unless you go through another subroutine, which does it constraint after constraint for every variable. So, there is a tradeoff between the number of steps versus the time you spend in a step. There is a trade off there. Conventional wisdom tells you not to bother about more than one constraint at a time and let the algorithm handle it with increased steps.

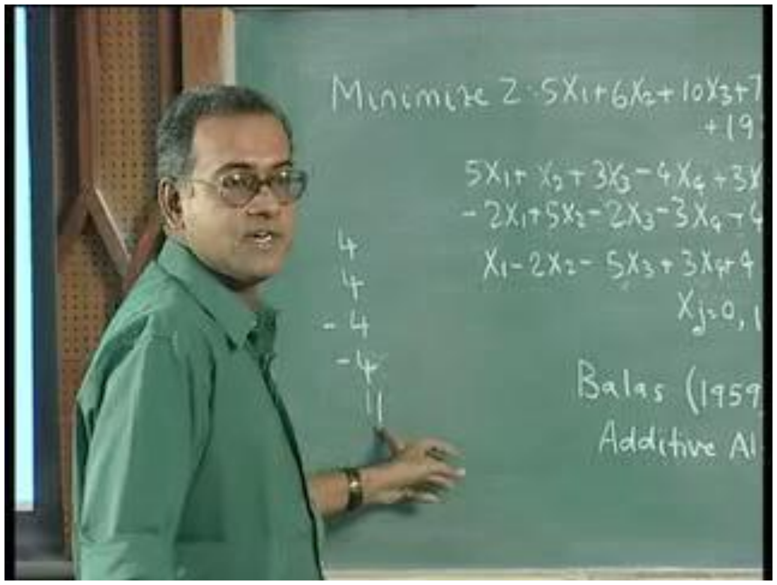
Number two, when we went here, I did move to step 11 by trying to enter something, but then we realised that we could backtrack here, because putting both the helpful variables cannot provide feasibility for a violated constraint. That is again something which you have to decide whether you want to consciously implement it in every iteration and verify whether - take every constraint and see - among the helpful variables which are helpful with respect to that constraint, try to put all those to 1 and see whether it is violated or not and then do it. That involves some more amount of CPU time and if you want to spend that CPU time in a step, you are welcome. Otherwise you can just leave out that second way of implicit enumeration through infeasibility. Let it take something and then it automatically it will show infeasibility after one or two steps. The only disadvantage is, the more you start putting them in, and fixing them to 1, ultimately you have to fix them to 0 by backtracking. It would involve more backtracks, more steps. So again there is a tradeoff between increased computational efforts per step versus more number of steps in the algorithm.

The third thing which has not come out very explicitly in this is how do I choose the helpful variable to enter? There are many ways of doing it. You have to remember all this, because you should assume that you are going to write a program to solve this, not solve it on the board or on the notebook. When you write a program to do that, then the easiest thing to do is to enter the first helpful variable and finish it. If you want to choose that helpful variable which contributes, then you need to write some more lines, which would take some more time to evaluate it. It will involve a sorting algorithm or trying to find the maximum in a given array and then you may have to store the extent of helpfulness with respect to the violated constraints separately, which needs just one more vector to be created and stored and so on.

Typically, just take the first one and blindly put it. Or what people suggested is, to have an index of helpfulness which is independent of the violated constraint. Every helpful index set comes and that is dependent on the set of violated constraints. What you try do is just keep an independent index of helpfulness, which simply means just add the coefficients column wise for all the variables. When you do that, the helpfulness for variable $X_1$ is 5 minus 2 plus 1 which is 4. It is just an index; that is all there is nothing special about it.

The extent of helpfulness here is 5 plus 1, 6 minus 2 is 4; extent of helpfulness here is minus 7 plus 3 is minus 4; this is plus 2; this is 4 plus 4, 8 plus 3, 11; this is minus 4 minus 3 is minus 7 plus 3 is minus 4.

(Refer Slide Time: 33:38)



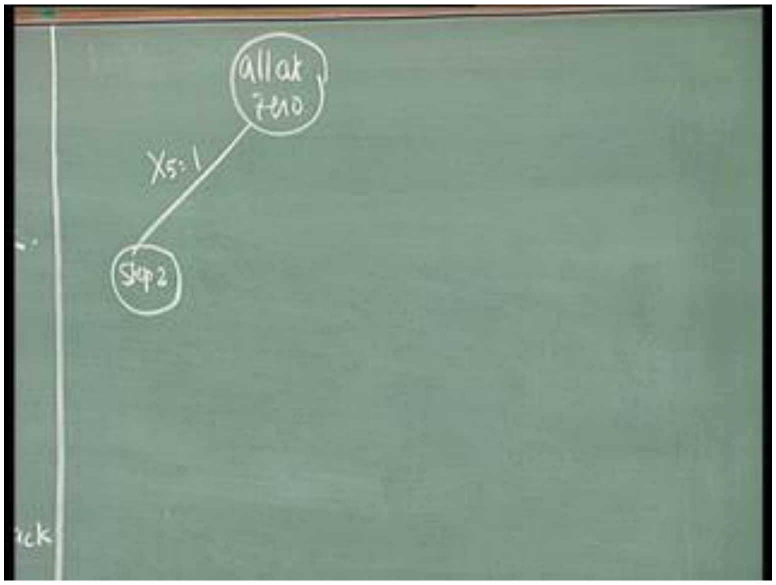
What you to try do is keep this index fixed. This index is independent of which iteration you are, where you are. If you want a simple thumb rule to tell you which one to choose, look at the set here and identify that variable which has the maximum of these. It is just a simple guideline that would have straight away taken you to $X_5$ in the first iteration. $X_5$ would have given you a feasible solution. This does not guarantee any reduction in the number of steps. It is just that, you know, experience and computation will tell you that this is not a bad rule to choose. Look at your set; a set has to be defined exactly the way it is defined. This set has to be dependent on the violated constraint, it will be different in different iterations, depending on V. Only when it comes to the issue of choosing, which among the helpful variable I want to put, now go back; instead of doing it constraint wise or violated constraint set wise, just look at this index and this will tell you, for example, that in general you can prefer to or choose to enter $X_5$ compared to any of those. You would like to enter $X_1$ or $X_2$ in preference to $X_3$ or $X_4$. There has been considerable research since 1959 on trying to make this algorithm better and better. There has to be a method, but right now, I am not aware of one which does this straight away; there has to be, one needs to go to the literature and see.

There are multiple ways of doing it. Again, what will happen is, you can also go back and show that under certain conditions or for certain problems, certain thumb rules may work better. The bottom line is it is still a worst case complete enumeration algorithm; that is something which you cannot come out of. Depending on, for example, the nature of the constraint or the sparseness of the matrix, one could do many things and people have done many things.

The implementation that I have given is an old 1959 implementation. Lots of refinements have happened to this, in terms of choosing the helpful variables. Nevertheless, even the first and the old implementation is a very good one, because all you need is just three vectors. If you have a problem with m constraints and n variables, all I need is S which is always going to be a vector of length n, as many variables, this is as many constraints, this is as many variables. That is all you need. All you need is 2m plus n, you do not need anything more, plus one place to store the solution, which could be another n and a number; that is all you need, which is abysmally minimal amount of memory required to solve a problem of this kind, no matter what the size is; just 2m plus n, not more than that.

Now what is this backtracking and what are we trying to do in this backtracking. That is the next thing that we need to know. What is this bar business, why should I go from the right, why should I leave out something to the right? All these are questions that we have to look at. So let us look at that question.

(Refer Slide Time: 38:05)

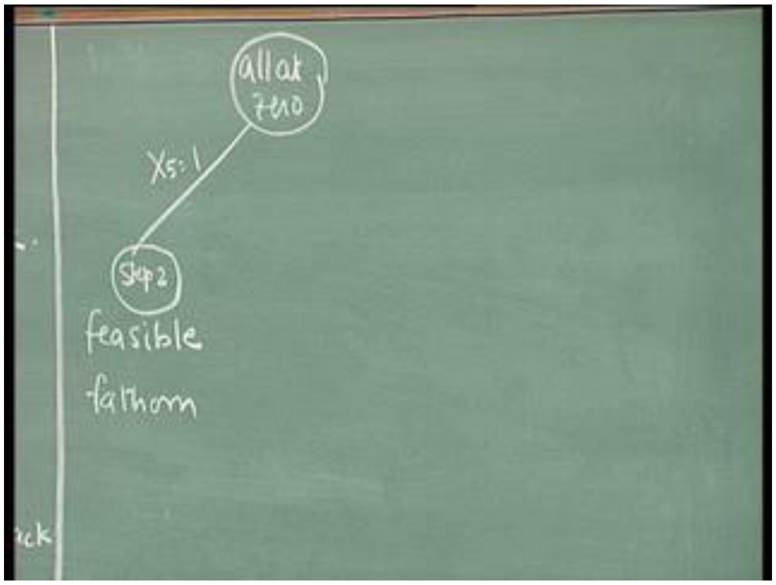


Let us try to draw what is called a branch and bound tree associated with this solution and everything will be clear. First, we said all of them are at 0, but not fixed.

The first step $S_1$ was this, was the first node which is shown there. Now here, this 5 implies $X_5$ is fixed at 1. So basically I am trying to branch from this on variable $X_5$ and I say here that I have created an $X_5$ equal to 1. The other branch could be $X_5$ equal to 0, which I have not created right now, but I know that such a branch exists because variable $X_5$ can take only two values, which is 0 or 1. When I branch on $X_5$, I create two nodes, one which fixes $X_5$ at 1, and the other which fixes $X_5$ at 0. I have not written the second one. So this is $X_5$ equal to 1, this is your solution at the end of step 2.

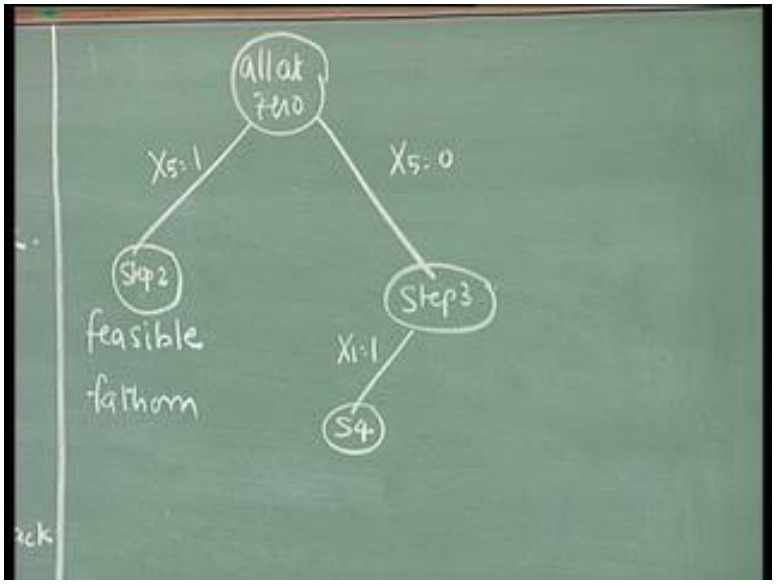
Now I come to step 2, this is the solution.

(Refer Slide Time: 39:18)



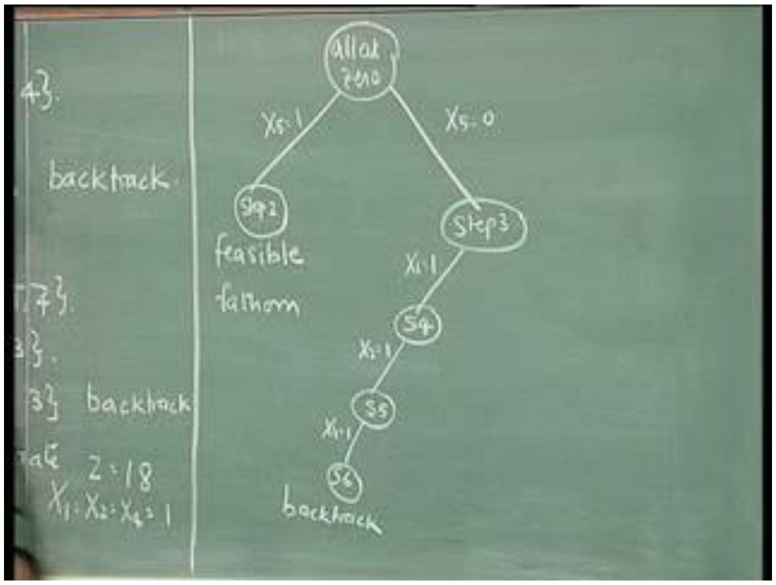
This is feasible and please remember that when I branch from here, with $X_5$ equal to 1 and $X_5$ equal to 0, there are 32 possible solutions; 2 to the power n possible solutions. When I branch on a single variable, there are 16 on this side and there are 16 on the other side. I have evaluated only 1 out of this 16. Because of this feasibility, I am implicitly enumerating the remaining 15, which was the question asked at that time and the answer is here. I am implicitly enumerating the remaining 15, because I have got a feasible solution. I know from my rule 1, that the moment I have a feasible solution, fixing those variables that are at 0 to 1, cannot give better solution. This part is fathomed. You say, you fathom; you do not move further, you fathom by feasibility and implicitly evaluate 15 other solutions. I have already evaluated 1. Now, what should I do? I cannot proceed further here because I have fathomed this. I have to go on the other side. To go on the other side, I have to create a node with $X_5$ equal to 0 and that is exactly what I do, by putting a 5 bar. A 5 bar implies fixing $X_5$ to 0 and I very conveniently do it, by just going back and then I try to go as far back as I can and then create a branch to the right. I have already evaluated this node; I cannot create a branch to the right from this. I do not want to create because this is fathomed; so I move 1 step above. From here I can create a node, so I create this. This is $X_5$ equal to 0 and this is your step 3.

(Refer Slide Time: 40:29)



You go to step 3, you have this 5 bar. Now your step 4, you are branching on variable $X_1$ from here. I have already realized that I am not going to look at the left hand side anymore, because there are 16 possible solutions. It turned out the first one is feasible, so the remaining 15 are implicitly enumerated. So I go back to the right and then I look at this 1, means we are branching on $X_1$ equal to 1, which is my $S_4$. I call it as $S_4$ which is your step 4.

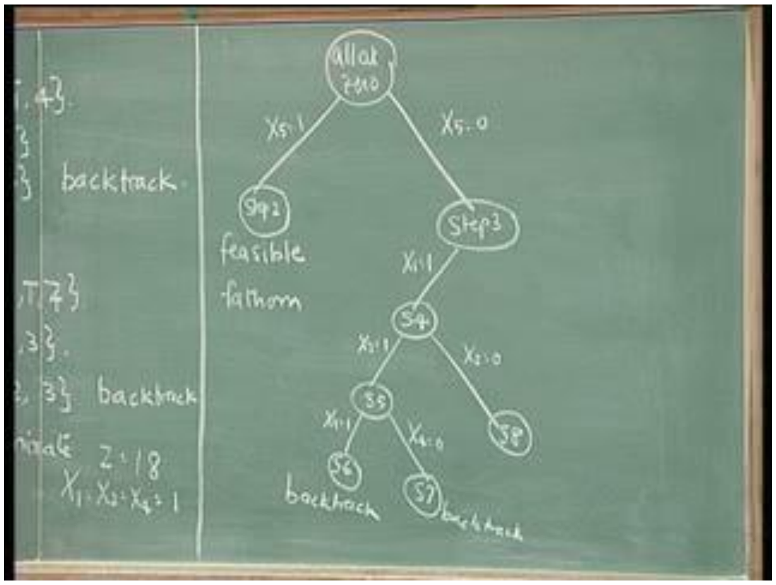
(Refer Slide Time: 42:10)



Right now I do not branch the $X_1$ equal to 0. I just leave it as such. Now this is what I have. Now again we go back and branch here, on $X_2$ equal to 1, which is my $S_5$. Again, I can branch on the right, which I do not do now. Now I go here and define an $S_6$ where I am branching on $X_4$ equal to 1. Again I can create a branch on the right, which I have not done. This I backtrack, I fathom by feasibility, because V is a null set. Set of violated constraints is a null set, which means I a feasible solution; I fathom by feasibility.

It turns out here, that out of these 5 variables, 3 variables are at 1; 2 variables are remaining, but this is already fixed at 0, so 4 variables have been fixed. There is only 1 variable that is left to be fixed, so I am implicitly evaluating only one more solution. There are two possible solutions from there; I have got that feasibility, so I can implicitly evaluate only one more. I have implicitly evaluated the solution with $X_3$ equal to 1. That is the solution which I am not evaluating. So I backtrack. Whenever I backtrack, I implicitly evaluate something; so I backtrack from here.

(Refer Slide Time: 44:01)



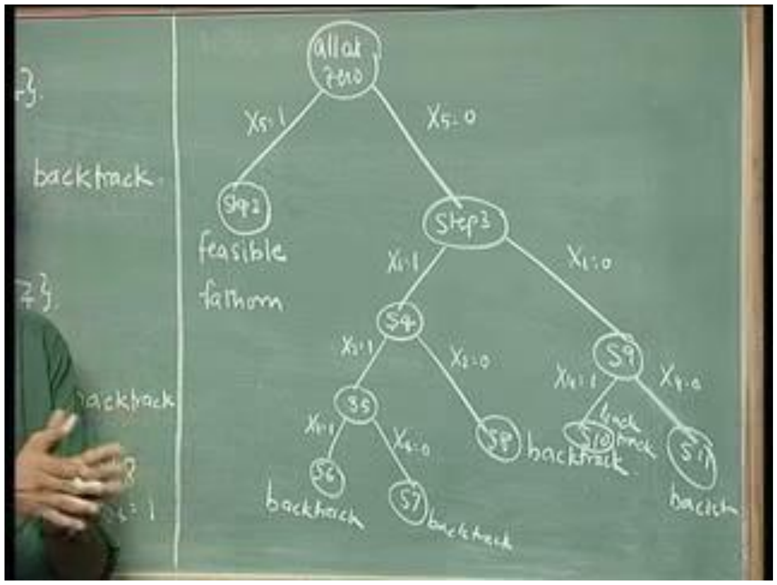
When I backtrack from here, what I actually do is something, but conventionally what I would have done is, I go back here and I would have created this branch. That is what I would have done conventionally. I do not do that, what I do is, I branch from a very convenient place. Once we backtrack, I move up; this is a place where one more branch can be created. So I just do that, so I create the branch $X_4$ equal to 0 which is my $S_7$. So my $S_7$ comes here with this. Now, 1 and 2 remain here, because as far as this branch is concerned $X_5$ equal to 0, $X_1$ equal to 1, $X_2$ equal to 1, $X_4$ equal to 0, so this remains, 4 becomes 4 bar. Here, I fathom by infeasibility, because I backtrack because of T (Refer Slide Time: 43:09). Whenever I backtrack because of T, I have a V and then T is a null set, I fathom by infeasibility. When V is a null set and T does not exist, I fathom by feasibility. I fathom by infeasibility, again I have 4 variables fixed. There are only two solutions possible, one of which is already this. So I implicitly evaluate one solution only through this, so I backtrack from here. Again, conventionally I would like to backtrack from this because this is available and this is closer to the root of the tree. But then I do not do that, I just go back. This is where I go; I go one step back. Now I realize I can actually branch from this. I create the branch $X_2$ equal to 0, I go to $S_8$.

Now go back to $S_8$, I have 5 bar, 1, and 2. Please look at it carefully, what did we do here; when we backtracked on 2, we left out this 4. Why do we do that? That is because you want to come from here to this. I have the freedom again to branch on 4 if I want, so that

I get because of this. That is why you leave out this, anything that has got a bar. When I come from the right, I backtrack on 2, I do not retain this bar, I am leaving this variable; I can look at it again. Now this would mean I am fixing only 3, because I am doing this at this level. That is the reason why, when I come back here, for example, I leave out that 4; I do not retain the 4. So anything to the right of the element that is backtracked will automatically go, which implies that you are moving one layer above and backtracking or I am creating an additional node. So, this is your $S_8$ which has $X_2$ equal to 0.

At $S_8$, I again go back and backtrack here, which means $T_8$ is equal to 4 (Refer Slide Time: 45:23). I backtrack on T which means I fathom by infeasibility. The logic here is putting the helpful variable cannot bring feasibility, so I backtrack with infeasibility. Now the 3 variables are fixed, 2 are remaining, 4 possible solutions; I have got one of them already. So I am evaluating 3 solutions implicitly. Now I go back and work backwards and then move above, I can create one more thing, so I go here and create $X_1$ equal to 0. Now let us see how that happens.

(Refer Slide Time: 48:27)



I backtrack here, this 2 goes, and I put a 1 bar. Whenever you move one level above, because of the backtracking, you will leave out the bars. So the 2 will go. Here I am fixing just 2 variables in this tree. I still cannot fix variable $X_2$, $X_4$ and $X_3$ in many ways.

So I come to this, which is my $S_9$. If I go to $S_9$, I continue again to $S_{10}$ by putting a 4, so I branch on $X_4$. So $S_4$ equal to 1 is what I have, again I backtrack. So this is my $S_{10}$, I again backtrack. Here again I backtrack on infeasibility, because there is a T that exists, and I still backtrack, which means I am backtracking on infeasibility (Refer Slide Time: 46:52).

There are 3 variables fixed, 2 are possible - $X_2$ and $X_3$, 4 solutions are possible. This is 1 out of the 4 possible solutions, remaining 3 you evaluate by implicit enumeration. I backtrack here; I go back, I can create something. So I create this $X_4$ equal to 0 and get my $S_{11}$. Now in this case, what will happen is, the 4 will become 4 bar, nothing else will happen. When you move from a 0, 1 back and then one level above, you will have this. Whenever you move one level above, you will have a right hand side going. If at the same level you are backtracking it will not happen.

$S_{11}$ is 5 bar, 1 bar, 4 bar; $X_5$, $X_1$, $X_4$. Now you realize, again you want to backtrack by infeasibility. Now 3 values are fixed, 2 variables are remaining, can be done in 4 ways, out of which one of the solutions is this. So 3 more solutions you have implicitly evaluated. Again you want to backtrack, but no backtracking is possible; therefore the algorithm terminates. So out of the 32 possible solutions, this has evaluated 11 possible solutions and the remaining 21 have been implicitly enumerated. The reason why you are able to do this, only with 3 sets or 3 vectors, is because of the order in which you backtrack.

This algorithm forces you to move in a certain order. If the algorithm tells you, that if you choose to move in a certain order then, you do not have to store everything as independent or individual nodes; you could rather store the whole thing only as 3 different vectors and proceed. It is a branch and bound tree, but it is a very efficient way of representing a branch and bound tree. We look at other aspects of zero-one in the next class.