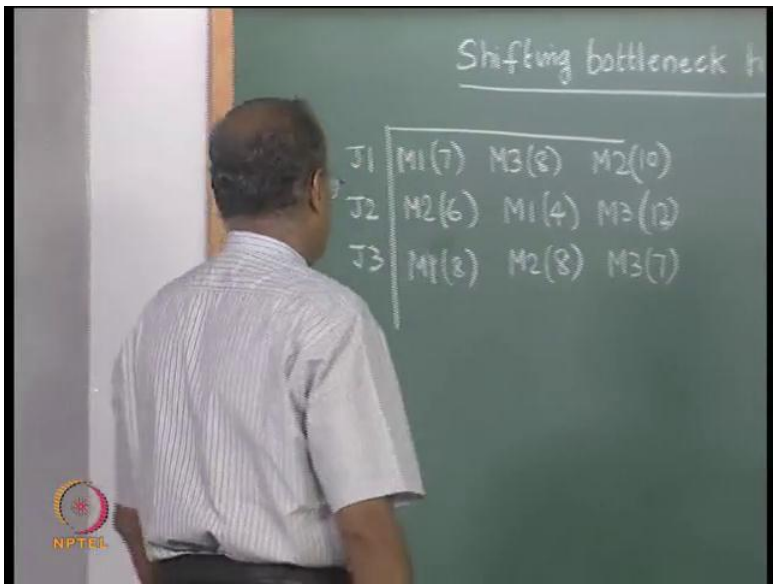
Operations and Supply Chain Management
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

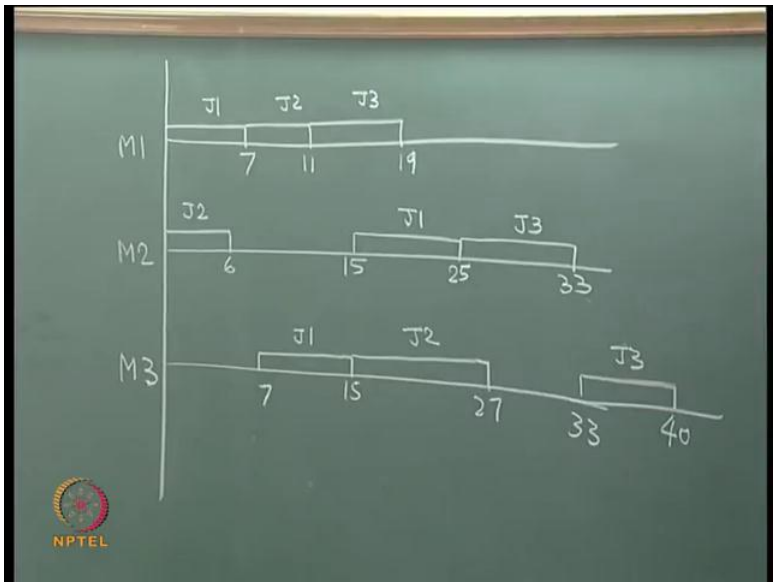
Lecture - 29
Job Shop Scheduling – Shifting Bottleneck Heuristic

(Refer Slide Time: 00:17)



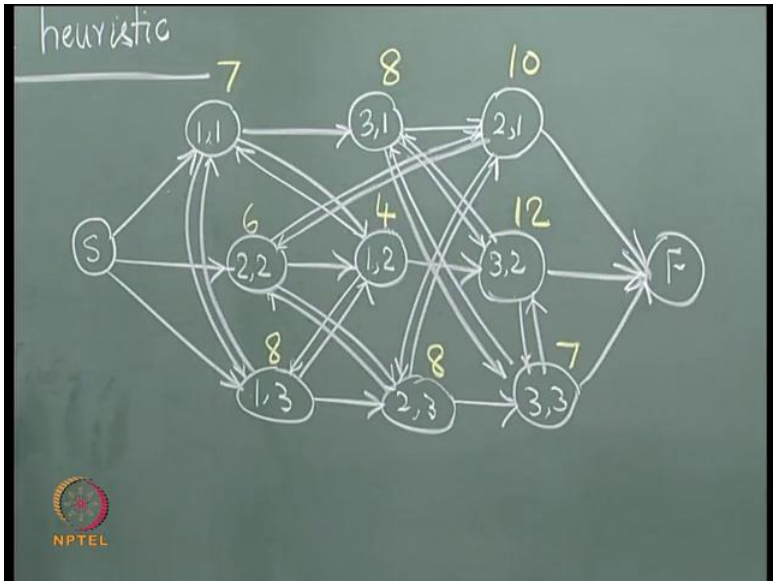
In today's lecture, we will look at the Shifting Bottleneck Heuristic, to try and minimize the Makespan in the job shop scheduling scenario. In the previous lecture, we had found a solution using the SPT rule, which gave us a Makespan of 40.

(Refer Slide Time: 00:27)



Now, we will apply the shifting bottleneck heuristic to the same example, to see whether we can get a better solution or to understand, how the shifting bottleneck heuristic works. Now, in order to understand the shifting bottleneck heuristic, let us first draw a network, which represents the job shop scheduling problem.

(Refer Slide Time: 00:57)



Now, let us look at this network, this network captures a part of the information that is given here. Now, there is a start and there is a finish. Instead of looking at the job shop scheduling problem through a Gantt chart, which is what we did in the previous lecture,

let us try and look at it through this network. So, there are three jobs to be done, and these three jobs represent three paths in this network, so this would represent the job 1 first visits machine 1, if we see the route of job 1 it is machine 1, machine 3 and machine 2.

So, job 1 on machine 1 followed by machine 3 followed by machine 2. So if we trace the path of job 1, then the arrow will have to be like this. Similarly, job 2 first goes to machine 2, machine 1 and machine 3. So job 2 first goes to machine 2 machine 1 and machine 3, and therefore the directions for job 2, flow of job 2 will be like this, and similarly for job 3 it will be like this.

We also have a certain processing time associated with each node, and each node represents an operation which is the visit of a job on a machine. So, if a node has a i comma j, then j is the job and i is the machine, so 1, 1 has value 7. This has 8, this has 10 6, 4 and 12, and this is 8, 8 and 7. So, left to itself, each job will go through the machines. The order of visit of these machines are captured, but then we also have a condition that a machine can process only one job at a time. So, if we take a particular machine, now the machine figures here, machine 1 figures here as well as machine 1 figures here.

So, machine 1 would now take the jobs according to a particular sequence, and depending on that the time taken will change. So, if we take machine 1 and show this we could have for example, between jobs 1 and 2 we could have something like this, where at the moment we do not know, whether job 1 is going to go first on machine 1 or job 2 is going to go first on machine 1. So, if job 1 goes first, and then job 2 goes on machine 1 then something like job 1 will go first then job 2 will go on machine 1, and the flow will continue.

If job 2 is going first and job 1 is coming later, it is like saying job 2 will do it first, then the arrow will move towards job 1 and then it will proceed. So, at the moment we do not know that if we consider these two jobs on this particular machine, we do not know the order in which they are going to be processed by the machine. And depending on the order we know that one of these two will be active, while the other will not be active, similarly we have to consider for machine 1 jobs 2 and 3, and also for machine 1 jobs 1 and 3.

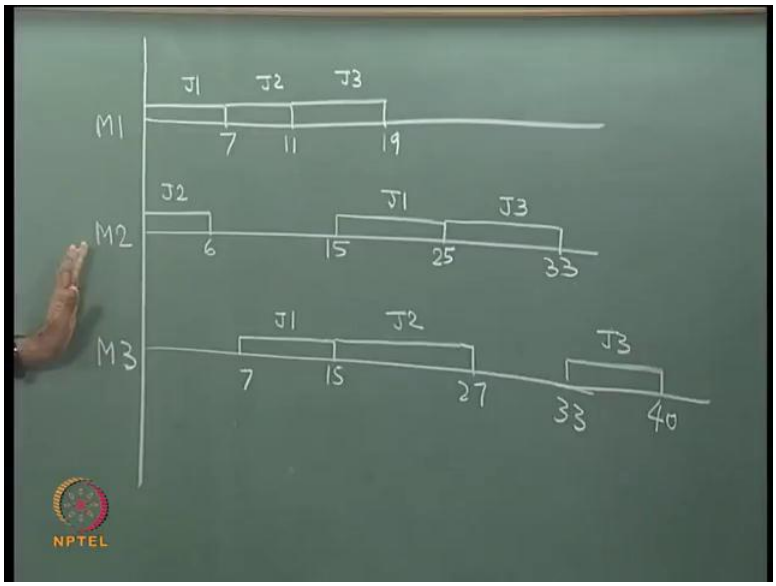
So, there are actually for every pair of jobs there are two arcs, one of them will finally be active depending on the solution, similarly we have to do for machine 2. So, machine 2 is here machine 2 is here, and machine 2 is here, so we would have something like this, something like this and something like this, depending on the order in which the machine is going to process the jobs.

As I mentioned, for every pair in the final solution, only one of the arcs will be active, now similarly for machine 3 there are, machine 3 figures in 3 different places here, so we will have one set like this, we will have one set like this, and we will have the third set like this. So, this completes the network, where these type of arcs will not change the directions will not change, and in all the places where we have two arcs, depending on the solution only one of them will be active in the final solution.

Now, if we draw such a network, and our decision now is to find out if we take machine 1 out of these 3 sets of arcs, 6 arcs that we have written, which of them are going to be active and which of them are not going to be active. To put it differently, when we decide which of them are going to be active, and which of them are not going to be active, we automatically are deciding the order or sequence in which this particular machine is going to take up the jobs.

So, if we are able to get the set of active arcs out of these bunched arcs, in an intelligent manner such that the longest path is reduced, then the Makespan is automatically reduced, the longest path on this arc indicates the Makespan of a given feasible schedule. Let us explain that aspect by first looking at the SPT solution showing, how we are able to capture that 40 on this network, and then showing how the longest path represents the Makespan of a given schedule.

(Refer Slide Time: 08:22)

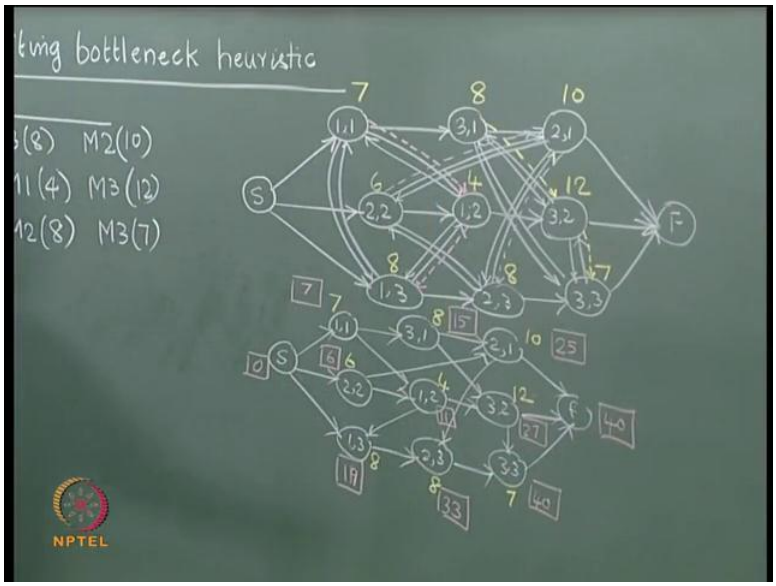


So, we have already seen this Gantt chart or bar chart in a previous lecture, where the Makespan is 40. More importantly for the SPT schedule, M 1 is taking the jobs in the sequence J 1, J 2 and J 3. So, let us try and match that here were M 1 is taking them in order J 1, J 2 and J 3, so M 1 is first taking it in this order, so this is done first, then this is done, then this is done, so M 1 is taking first J 1 then J 2 then J 3.

So, let me just show this with a different color, and put a dash here to indicate that out of these two arcs that are connecting these two nodes, this arc this particular arc with an arrow mark here, indicates that J 1 is done first and then J 2 is done on M 1. Similarly, since 3 is done after 2 between these two arcs, now this arc will be active the arrow mark here will be active, which is now shown by this dotted arc.

So, for the SPT sequence on M 1, the two dotted arcs are now replacing this pair, this pair, and this pair as far as this pair is concerned both of them are inactive, because we will not put 1 to 3 the sequence in, which it happens is J 1, and then J 2, and then J 3. So, automatically when we put one arc here and another arc here it takes care of the fact that 3 is done after one and the route moves like this. So, both these will not be used, so only 2 out of the 6 will become active according to a feasible schedule.

(Refer Slide Time: 10:38)



Now, let us look at M 2 on the SPT chart and realize that the order is J 2, J 1 and then J 3. So on M 2 first it is doing J 2, then it is doing J 1 and then it is doing J 3. So, let us replace this with this arc here, and then this arc, which is this. Similarly on M 3 it is again J 1, J 2 and J 3. So, on M 3 it is first J 1, then it is J 2, and then it is J 3, so the order will be and then J 3. So, now, this network has become slightly clumsy, because I have overwritten the colored arrows, but then we have to find out the longest path on this network, considering only the colored arrows as well as these kind of arrows.

So, to do that quickly, let us draw this network again with the relevant arcs and then try to find out the longest path, so we have s 1 1, 3 - 1, 2 1, F - 2 2, 1 2, 3 2, 1 3, 2 3, 3 3 and F. Plus of course, 1 1 to 1 2, and then 1 3, then 3, 1, 2, 3, so 3, 1, 3, 2, 3, 3 and as far as 2 is concerned 2 1, 2 3, 2 2, 2 1 and 2 3, so 2 2, 2 1, 2 3. So, this is the arc that we have, now we have to find out the longest path on this, we also have to write down the process times 7, 8, 10, 6, 4, 12, 8, 8, and 7.

So, we will now start with from here to here, so this node the value is 7, now this node the value is 6, now the longest path to reach this node is 7 plus 4, and then you come down here, so the value is 11. And if we see very carefully these are exactly the values of 7 and 11, which represents the start times, so 0 is here, so you realize 0, 7 and 11 plus another 8 of course, will happen, which is 19. Then we have to move to this node, this is

15, 7 plus 8, 15, which also we can see in that chart, now we come here, so this is 11, 7 plus 4, 11.

Now, we have to write in such a manner, that we take it after we include this, so this will become 7 plus 4, 11. 11 plus this 8 will become 19. So, now, we have come up to this then in order to look at this, this will be 15 plus 10 25, this will be 6 plus 10 16, so we will look at this at 25. Now, we have evaluated this node, so this is 15 plus 12 27, now this will be 25 plus 8 33, and this will become 33 plus 7 40, and this is 0, so this will become 40.

So, now we realize that the longest path on this network has a length equal to 40, which happens to be the Makespan of the sequence, so we have essentially mapped the SPT solution, on to this arc to try and get the Makespan corresponding to the SPT solution. That is not what we intend to do, what we have tried to show is given a Gantt chart, from which the order of visit of the jobs can be ascertained, we have now mapped the Gantt chart on to this and chosen the relevant arcs corresponding to a given solution, and we have evaluated the Makespan to show both of these are the same, but that is not what we intend to do.

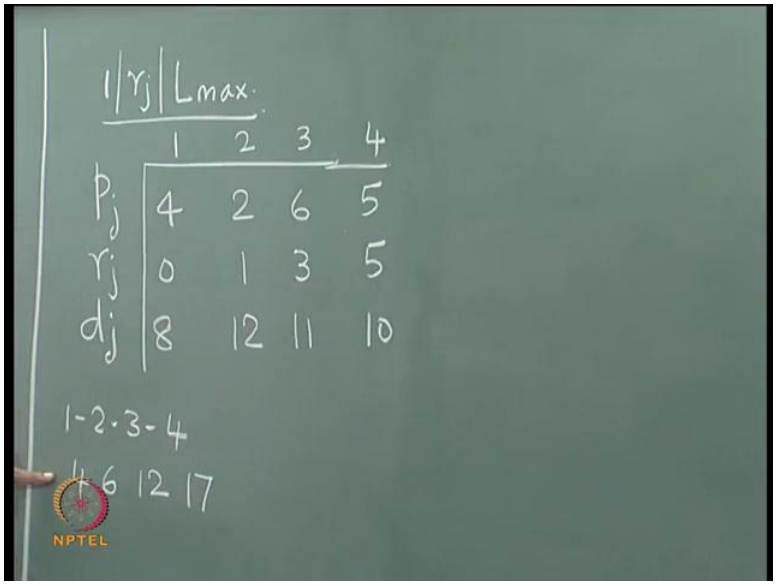
What we intend to do, is to try and find out what is the best way that we choose the 2 out of these 6. SPT made us choose this and this, now is there a another way by which we can choose, another set of 2 arcs out of this.

Similarly, another set of 2 arcs on the yellow and another 2 arcs on the other, which is machine 2, so that at the end of it the longest path is less than 40. So, if we are able to get the longest path as less than 40, then we have achieved a solution that has lesser makespan than the solution that has been shown here. The other learning also is that the important thing though is the computation of the makespan, and try to minimize the makespan, but the decision that has to be taken.

What SPT rule did is to try and find out a certain order in which the jobs are going to be processed on the machines, for every such given order there will be an makespan. So, what we try to do is we try to get that order by looking at these arcs, and try to find out what is a best arc that we use, each arc that we use represents an order in which the jobs are going to be processed on the machines.

So, let us now try and see now having understood this path, let us now try and see is there a best way or what is the way that we are going to adopt to choose the best 2 out of the 3. So, what I will do now is I will remove these, because these belong to SPT, so now we have to find out, if we take machine 1 out of these 6 arcs which 2 we are going to choose. Similarly, for machine 2 and similarly for machine 3, and once we have chosen the longest path will automatically give us a makespan. Now, in order to understand what is the best way to choose, these arcs we try and look at another sub problem, which I will explain. Now, and then use results from that sub problem to determine, what is the correct set of arcs that we need to choose, so let us go to the other sub problem.

(Refer Slide Time: 19:39)



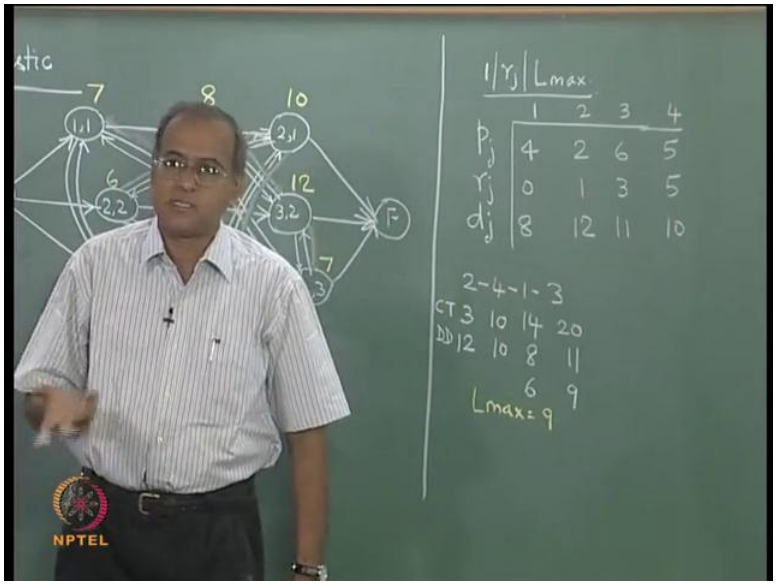
And this sub problem is called 1 r j L max problem, it means 1 stands for a single machine, r j stands for release times of jobs, and L max stands for maximum tardiness. So, the sub problem that we will look at is a single machine problem with release times, it means all time are not available at time all jobs are not available at time equal to 0, but the times in which the jobs are available is known. So, r j is the time at which job j is available, and L max is to minimize the maximum tardiness on this.

So, let us take an example, try to understand the 1 r j L max problem, how do we solve that and then see how we use results from 1 r j L max, and superimpose it on this network, so that we are able to get the correct set of arcs, that would try and give lesser makespan. So, let us take this example for 1 r j L max and let us look at problem like

this, where there are 4 jobs on a single machine. The processing times are 4, 2, 6, and 5, the release times are 0, 1, 3, and 5 and the due dates are 8, 12, 11, and 10, so if we consider a sequence 1, 2, 3, 4, then the completion times are 4 the job can enter at time 0.

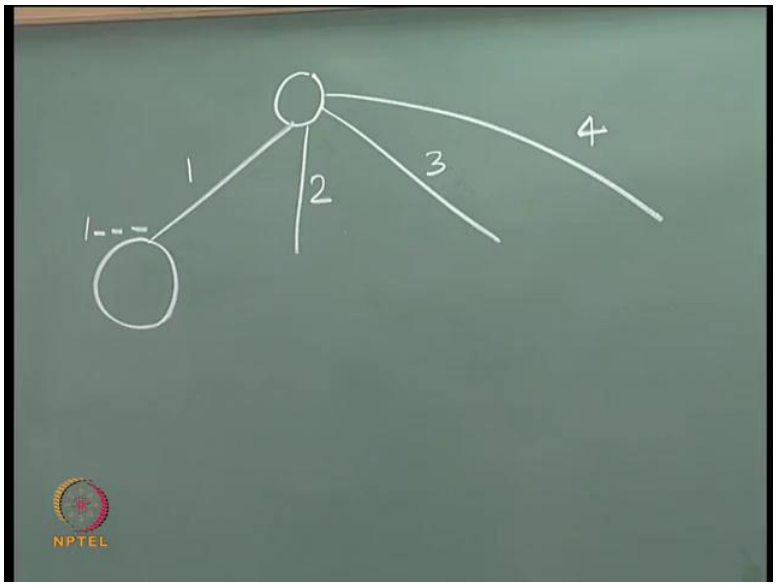
So, it can come out at 4 by which time job 2 is available, so 4 plus 2 6 job 3 is available once again at 3, so 6 plus 6, 12 and job 4 is available from 5, so 12 plus 5 17, these are the completion times 4 plus 2 6 plus 6, 12 plus 5, 17. We have not encountered a situation, where we have to the machine has to wait for the job arrival. That has not happened in this particular sequence.

(Refer Slide Time: 22:40)



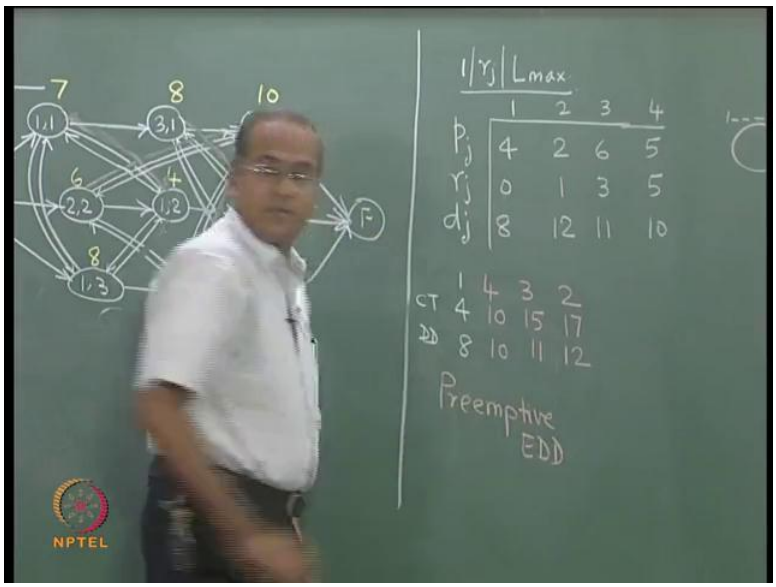
If we consider a sequence 2, 4, 1, 3 job 2 starts at 1, so finishes at 3, because it arrives at 1 takes 2 more units finishes at 3, 4 arrives at 5 takes 5 more units finishes at 10, 1 is available at 0. So, 10 plus 4 14, 3 is available at 3 14 plus 6 20, so these will be the completion times. Now, the due dates will be 12 here, for 4 it is 10, for 1 it is 8, for 3 it is 11, so these are completion times these are due dates, so this job is early, this job is neither early nor tardy, this job is tardy with tardiness equal to 6, this job is tardy with tardiness equal to 9. So, maximum tardiness is L max, so L max is equal to 9, there are 4 jobs in the worst case you can have 4 factorial sequences, so L max associated with the sequence 2, 4, 1, 3 is 9 what is the sequence that minimizes the L max is the problem.

(Refer Slide Time: 24:31)



So, now let us look at this problem and then go back and see what we do, so let us apply a branch and bound algorithm to do that. So, let us start here with the starting tree, and then let us say the job 1 is the first job, job 2 is the second first job, job 3 as first job, and job 4 as first job. Now, when we take job 1 as first job, we want to find out a lower bound on L max, so this can be written as 1 dash dash dash or 1 star star star, where 3 other jobs will have to be now put in a certain sequence. So, we now start with job 1 as the first job, so job 1 is ready at time equal to 4.

(Refer Slide Time: 25:17)



So, job 1 completion time is 4 due date is 8, now let us try to arrange the rest of the jobs using an earliest due date sequence. So, if we look at the rest of the jobs using an earliest due date sequence, then the tentative sequence will be 4, then 3, and then 2, that is a tentative sequence, if we use earliest due date for the remaining jobs. Now, the tentative completion times tentative completion times will be machine is available at 4, but this is going to be there it is going to come only at 5.

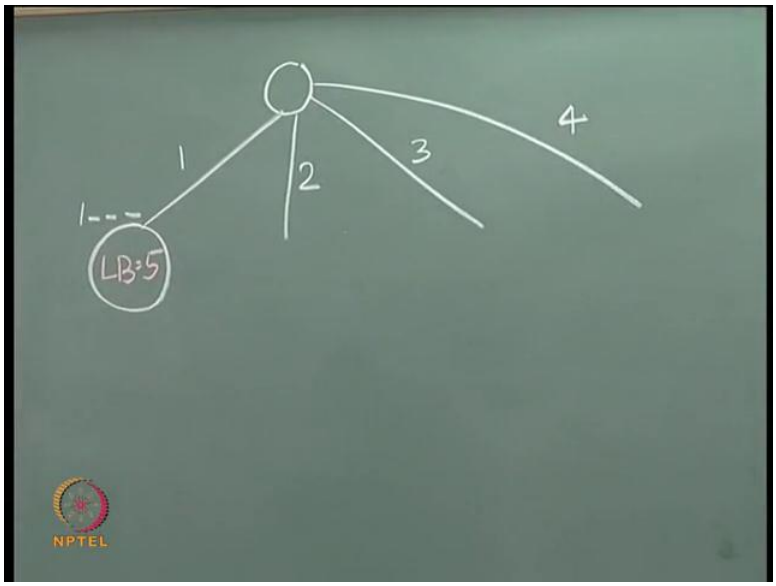
So, if we want to find out the completion time, then the completion time will become 5 plus 5 10, but we are interested in finding a lower bound to L max, therefore we will use what is called a preemptive EDD rule. Now, in the actual sequence, if we want to find the actual sequence for 4, it will start at 5 and finish at 10, and the machine is idle for that one time unit, because it is over at 4, but job 4 is available only at 5. Now, that additional 1, it can have an impact in increasing the maximum tardiness, but in a preemptive EDD rule, what we assume is that we allow job preemption.

So, we now see that if there is going to be a delay, because of a mismatch between this completion time and this arrival time, we look at the next available job or tentative job in the sequence, which is 3. Now, 3 is available at 3, so it is like saying I will do one minute of processing of job 3, take it out at time equal to 4, then I will enter this one and so this one will, now start at 5, and finish at 10, so completion time here will be 10.

Now, for 3 already I have done 1 unit of processing time, so now 3 will now go back and complete the remaining 5 units of processing time, finish at 15 by which time job 2 is available and then this will be 17. If on the other hand I had not used a preemptive EDD rule then this would still have been 10, but this would have been 16 and this would have been 18. So, preemptive EDD rule allows me to try to minimize my completion time only for the purpose of the lower bound computation.

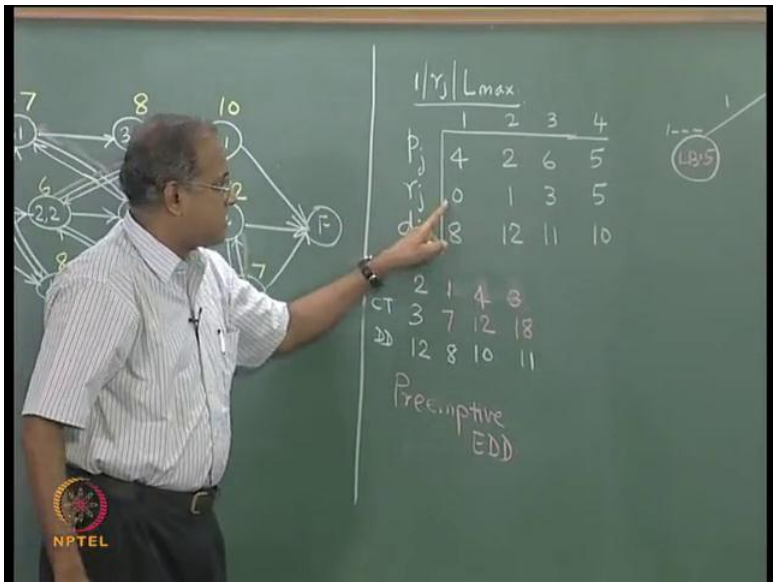
When I am actually evaluating L max for a given sequence. Then I will not use a preemptive rule, because one of my assumptions is that job preemption is not allowed, only for the purpose of computing a lower bound, I can use a preemptive EDD rule and allow preemption. So, that the lower estimate of L max is completed, a lower bound is a lower estimate of L max is completed, now due dates are 8, 10, 11, and 12, so this has maximum tardiness, so lower bound on L max is equal to 5.

(Refer Slide Time: 29:41)



Now, let me explain the computation of the other one, now we start with job 2 as first.

(Refer Slide Time: 29:57)



Here, we have to fix you cannot for the fixed jobs, you cannot apply the preemptive EDD, you can apply the preemptive EDD, only for the jobs that are not fixed in the partial sequence. So, 2 is fixed in the partial sequence, so 2 starts at n equal to 1 and finishes at time equal to 3. The next one will be 1, the next 3 jobs will be 1 3, 1 4 and 3 will be the next 3 jobs, which are not shown in white, at time equal to 3 jobs is available,

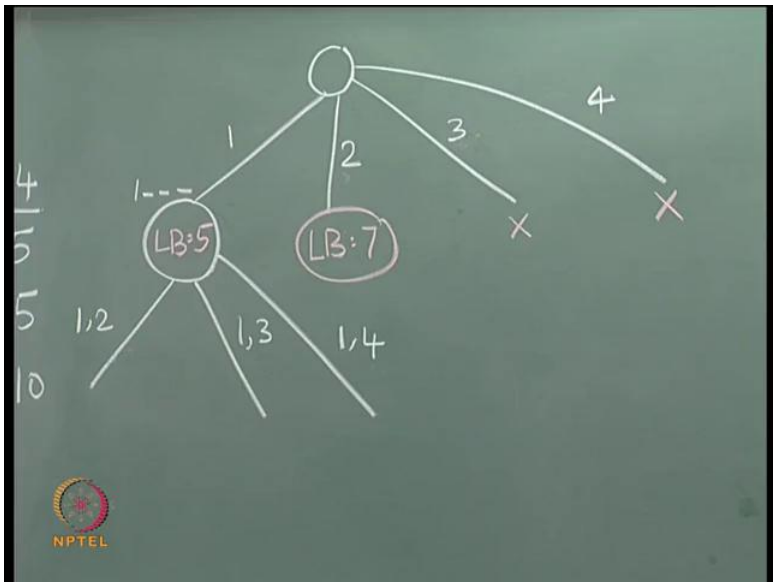
so this will be finished at 7, at 7 job 4 is available, so this will be at 12, and at 12 job 3 is available this will be at 18.

Now, the due dates are 12 for this 8 10 and 11, so this is going to contribute to L max, so L max is equal to 7, so lower bound is equal to 7. Now, here we have to look at can job 3 be the first job in the sequence, now if we see this problem very carefully, if job 3 is the first job in the sequence. Then definitely job 3 can start at 3 finish at 3 plus 6 equal to 9, and for 3 minutes machine has to be idle, but then in those 3 minutes idle, if I could easily have put job 2 in, and taken it out.

So, typically is there is a job here, as the first job 3 and there is some other job, which can be completed before this job can be started as the first job. Then the L max associated with this will always be higher than the L max associated with this, because this can be done ahead. So, see carefully if I have to do job 3 first, I have to wait for time equal to 3, so 3 minutes the machine is idle, and then after that I have to do 2, but common sense tells me that in those 3 minutes I can finish 2, because 2 starts at 1 and finishes at 3.

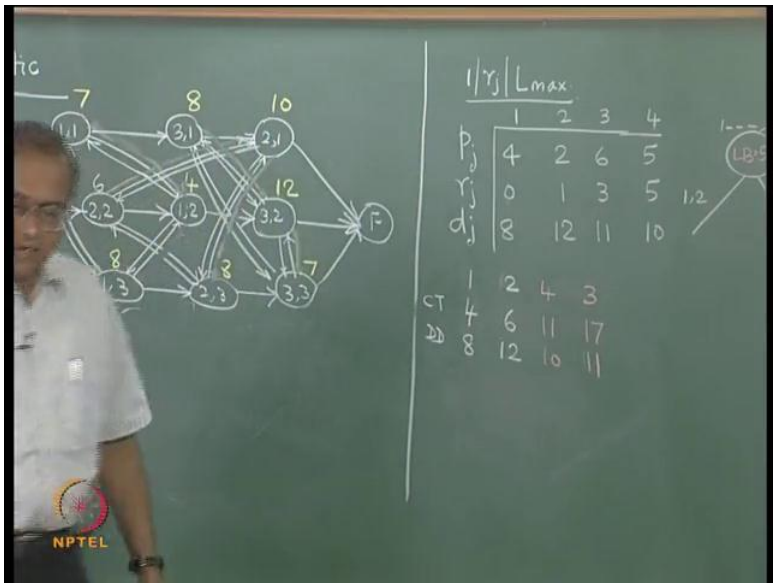
So, if there is a job like J 3, whose start is going to be delayed and in that delay, if I can finish some other job completely, then the lower bound here will always be more than the lower bound here. So, I do not calculate this at the moment I simply keep this vacant, because this is enough for me, so I do not do this. Similarly, for 4, because for 4 I have to wait for 5 more minute, in which time I could finish as well as I could have finished this, so I will not do for a 4.

(Refer Slide Time: 33:15)



So, now, I branch again on here, I branch with 1 2, 1 3, and I can actually branch on 1 4, so I could branch on 1 4 also, so let me just keep 1 4 also here. Now, we can continue this idea of the preemptive EDD, let me just explain the preemptive EDD for 1 2.

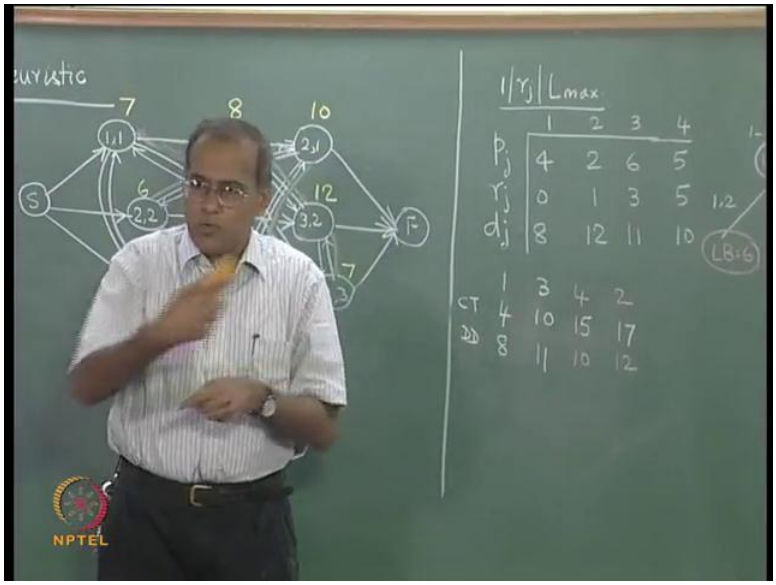
(Refer Slide Time: 33:51)



So, I have 1 and 2 which are fixed, so I completion time is I start at 0, I finish at 4 by which time this is available. So, 4 plus 2 6 now based on a preemptive EDD I will do 4 and 3, because the remaining due dates are 11 and 10, so I will do 4 and 3, now at time

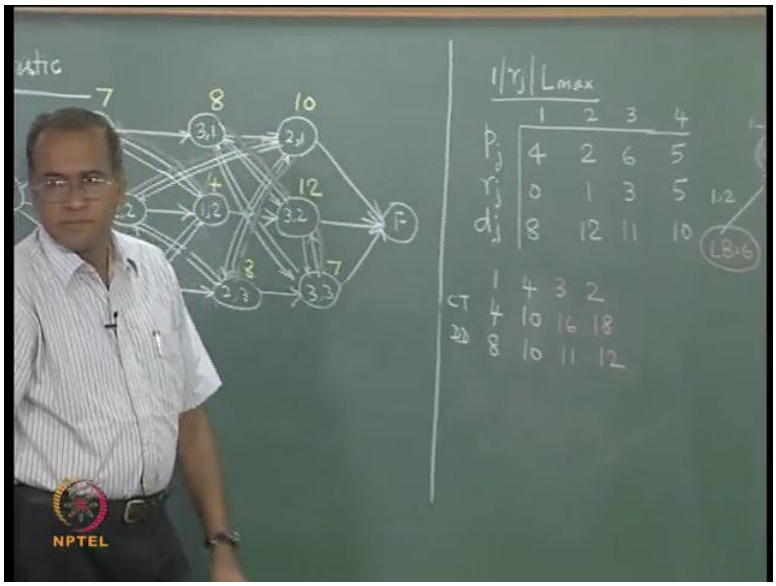
equal to 6 this is available, so this will become 11. And then this will become 17, so the due dates are for 1 and 2, 8, 12, 10, and 11, so lower bound becomes 6.

(Refer Slide Time: 35:00)



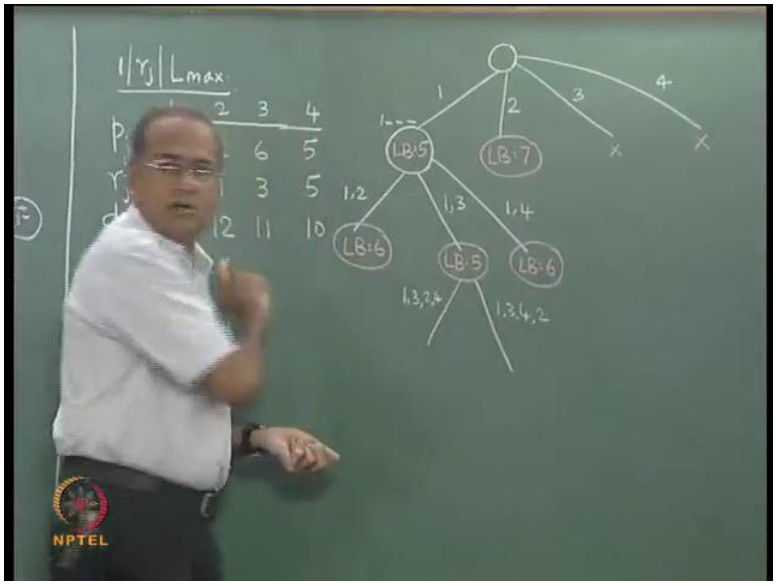
Now, if I do 1 and 3, which is this node, so at time equal to 4 this is available, so 4 plus 6 10, now these are the 2 remaining things, so I have 4 and 2. So, at 10 job 4 is available, so this will become 10 plus 5, 15, at 15 job 2 is available, so this will become 17. Now, the due dates are 8, 11, 10 and 12, so L max remains at 5, lower bound remains at 5. Now, there is actually a way by which we need not do 1 by 4, but let us try and do 1 4, and understand may be we need not do it.

(Refer Slide Time: 36:00)



So, we look at 1 4 here, now when we do 1 4 this is at time equal to 4, so this has to wait for another 1 minute, so takes it at 5 and finishes at 10. Then by EDD rule, I have 3 and 2, so at 10 this is available, so 16 and 18, now the due dates are 8, 10, 11 and 12, 11 for 3 12 for 2, L max is here, so L max is 6, so lower bound is equal to 6.

(Refer Slide Time: 36:50)

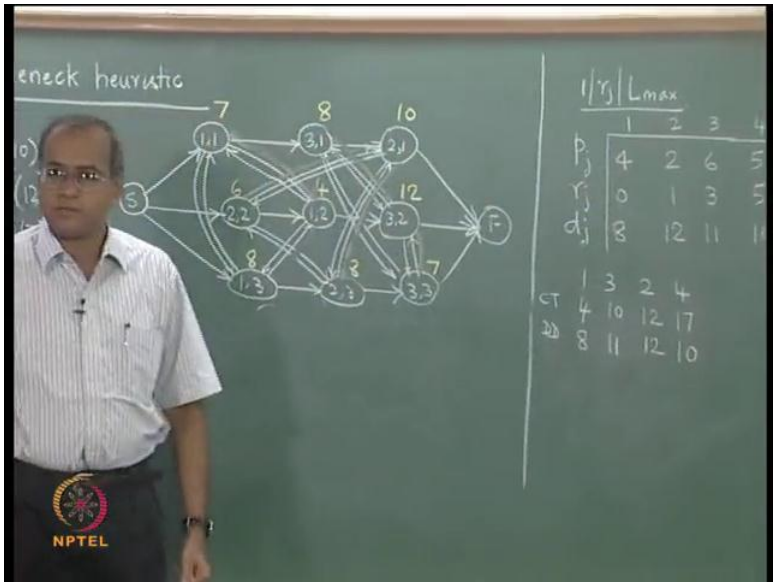


It is actually possible to show that we need not calculate this, because we have a 5 and this is going to result in at least one more, but in any case we have calculated this, so we keep this lower bound equal to 6. Now, we can proceed from here, and create 2 branches,

so I have 1 3 2 4 and 1 3 4 2, these are the 2 sequences that I can think of, because there are only 2 more jobs remaining, so these are feasible sequences.

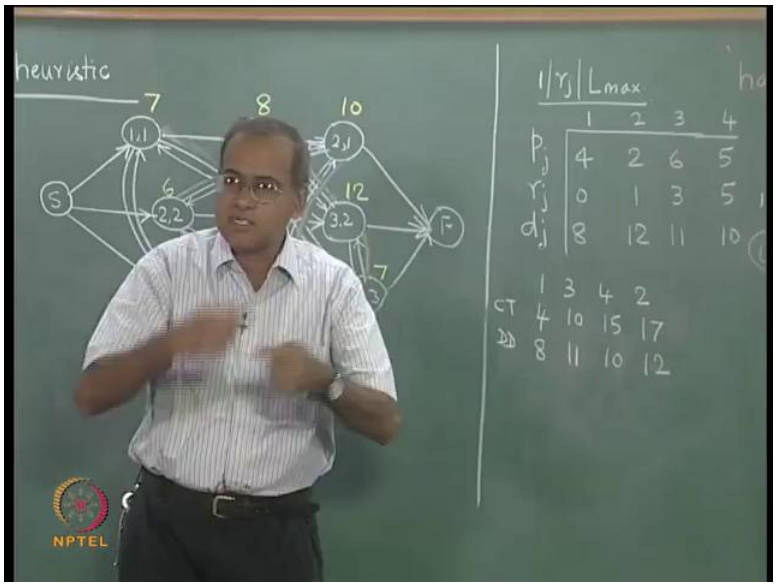
So, in general one would say even here we need not have evaluated this lower bound, we could have straight away said at 1 3, there are only 2 more jobs remaining, which can be fixed in 2 ways, so we could create 2 more sequences. Now, remember these are full sequences, so we will not use the preemptive EDD rule, we will simply evaluate the L max for these 2 sequences, assuming that job preemption is not allowed.

(Refer Slide Time: 38:10)



So, let us evaluate L max for these 2 sequences, so the first sequence is 1 3 2 4, so one completion time is 4, 3 is available 4 plus 6 10, at 10 2 is available 10 plus 2 12, and at 12 4 is available 12 plus 5 17. Now, the due dates are 8, for 3 it is 11, for 2 it is 12, for 4 it is 10, so L max is equal to 7 for this sequence.

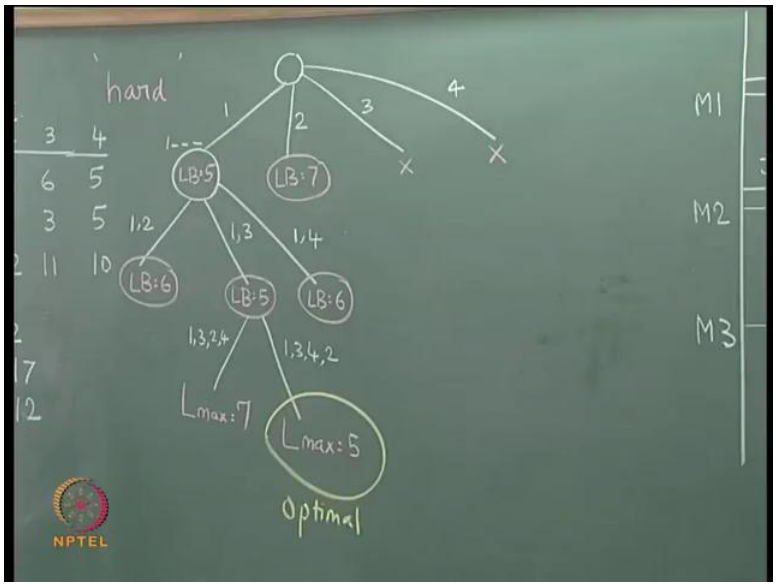
(Refer Slide Time: 39:08)



Now, let us look at the other one 1 3 4 2, so at time equal to 10, 4 is available, 15 at time equal to 15, 2 is available 15 plus 2 is 17, and the due dates are 10 and 12. So, L max is 5, L max is 5, L is 5, so L max is 5. So, we have L max equal to 5, and this is optimal or optimum, because with this 5, I can this is fathom by feasibility we cannot proceed. We have a feasible solution. This is fathomed by bound, we have already seen this idea in the branch and bound for flow shop sequencing. So here the lower bound is higher than the current best upper bound.

So, we need not proceed and in any case these 2 we will not do, therefore we can go back and say that 1 3 4 2 with L max equal to 5 is indeed optimum. So, this is a branch and bound algorithm to solve the 1 r j L max problem optimally, but then 1 r j L max problem is a hard problem. It is a difficult problem, and may in the worst case involve computation of all the n factorial possible sequences.

(Refer Slide Time: 41:01)



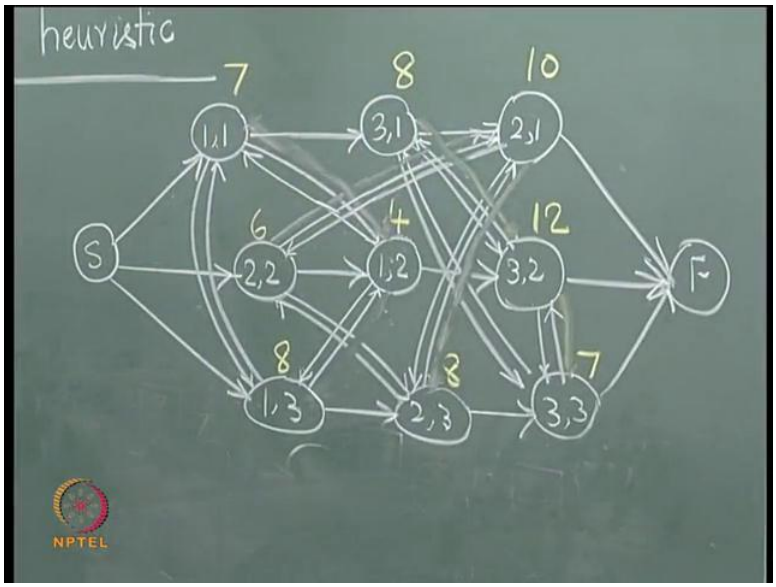
Now, we took an example with 4 jobs, so with these 4 jobs we may encounter a situation not in this case, but if there are n jobs, these 4 jobs could give us 4 factorial possible sequences, but then for another problem with n jobs. We may encounter a situation where we would be evaluating all the n factorial possible sequences, so 1 r j L max with the branch and bound can take us through an exponential number of computations, and it is a hard problem.

But, then using the preemptive EDD based lower bound which we demonstrated here, and also applying certain dominance conditions, which we explained through this example. The dominance condition ensured that we need not evaluate this or we need not evaluate this, simply because by the time this job is ready as the first job, some other job can be completed entirely, based on such a dominant rule, the branch and bound algorithm is found to perform very efficiently, even for a reasonable job size.

For example, with job sizes are like 30 or 40 or 50, one can write a very efficient code, which will solve this problem within a reasonable amount of computation time, when we do experiments or when we do an empirical style, though in the worst case 1 r j L max is harder. So, we still use this branch and bound algorithm to solve 1 r j L max, and try to get the optimum solution which shows the, which minimizes the maximum tardiness, on a single machine with release times.

Now, we have seen the 1 r j L max problem, and now we have to go back and use this 1 r j L max on the job shop scheduling problem, so that we try and get a good schedule at the end of the application of the shifting bottleneck heuristic.

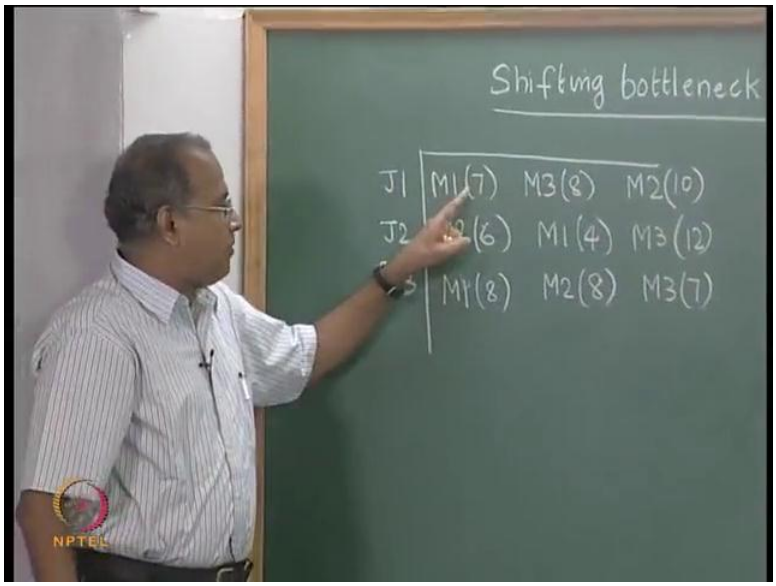
(Refer Slide Time: 43:22)



Now we will do that right now, so let us come back to this network, and then we understand that in this network all these arcs not going to change. But, wherever we have these paths for example, 1 1, 1 2 and 1 3 we said that out of these 6 arcs that are shown 2 of them will be in the solution, and which 2 will have to be decided.

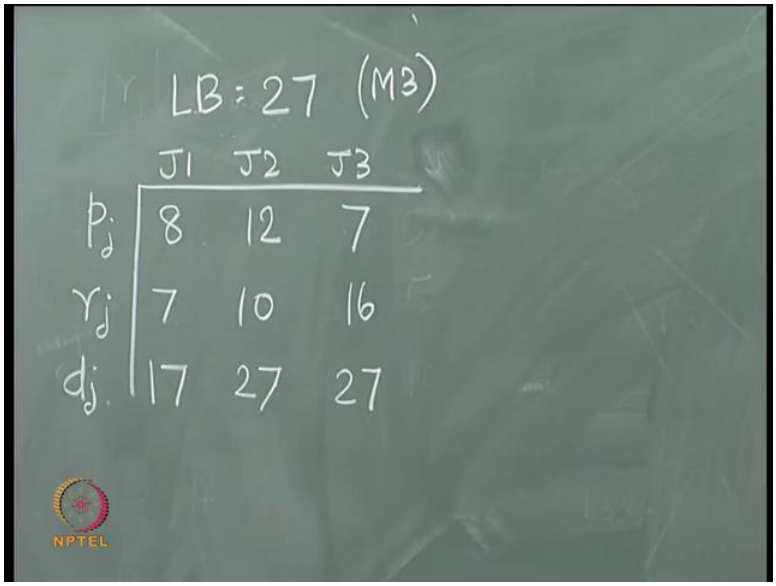
Now, let us look at this problem first ((Refer Time: 43:52)) and let us look at the load for each of these jobs, which means if I start looking at this path to completion, temporarily ignoring the these kind of arcs. If we start looking at only these paths, now this path is 7 plus 8, 15 plus 10, 25, which represents the total processing time for job 1 which is a simple addition of these 3.

(Refer Slide Time: 44:23)



So, job 1 requires 25 time units, job 2 requires 22 time units, job 3 requires 23 time units. So automatically the lower bound on the Makespan is the maximum of the job processing times, so 25 is a lower bound to the Makespan. Let us also try and look at machine times on the 3 machines that are required, so M 1 requires 7 plus 4, 11 plus 8, 19, M 2 requires 10 plus 6, 16 plus 8, 24, and M 3 requires 8 plus 12, 20 plus 7, 27. So M 3 with a load of 27 represents a slightly tighter lower bound on the Makespan. Now, we can say that since M 3 alone requires 27; obviously, the Makespan has to be greater or equal to 27, so we start with a lower bound on the Makespan as 27.

(Refer Slide Time: 45:28)



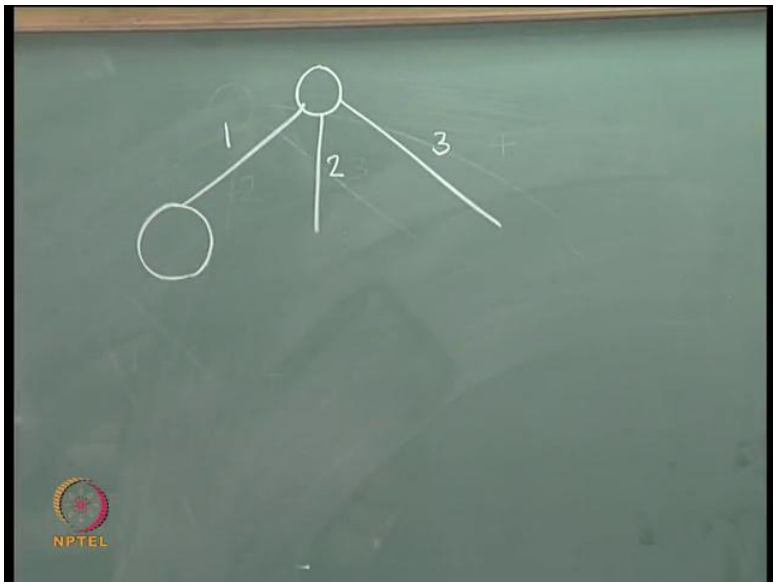
So, lower bound on the Makespan was 27, which is for M 3, and then we identify M 3 as the first bottleneck machine, because it has the highest load amongst the 3 machines. Now, we apply the 1 r j L max algorithm on machine M 3, and try to look at the 4 jobs with machine M 3 as the bottleneck machine. So, let us now write the 1 r j L max data for machine M 3, which J 1, J 2, J 3,- J 3, because we have only 3 jobs.

So, we have to write p j processing time on machine 3, r j release time or time at which the job is available, and d j which is the due date. Now, writing the processing times is easy, because we can see from here on M 3, J 1 requires 8, on M 3 J 2 requires 12, and on M 3 J 3 requires 7. Now, let us go back and find out the r j, which is the release time of the job or the earliest one can expect the job to be made available for M 3. So, if we look at J 1, J 1 earliest it can be made available for M 3 is 7, because it has to be finish one M 1, only then it has to come on M 3.

So, earliest release time for J 1 on M 3 is 7, earliest release time for J 2 on M 3 is 10, 6 plus 4, and earliest on J 3 again 8 plus 8 which is 16. Now, let us find out the due dates, if we assume that the Makespan is 27, which is the lower bound on it. Now, M 3 is the last machine for job J 2, so the due date job J 2 will be 27, which is the lower bound on the Makespan, similarly due date for this will also be 27, but now on M 1, M 3 is not the last machine.

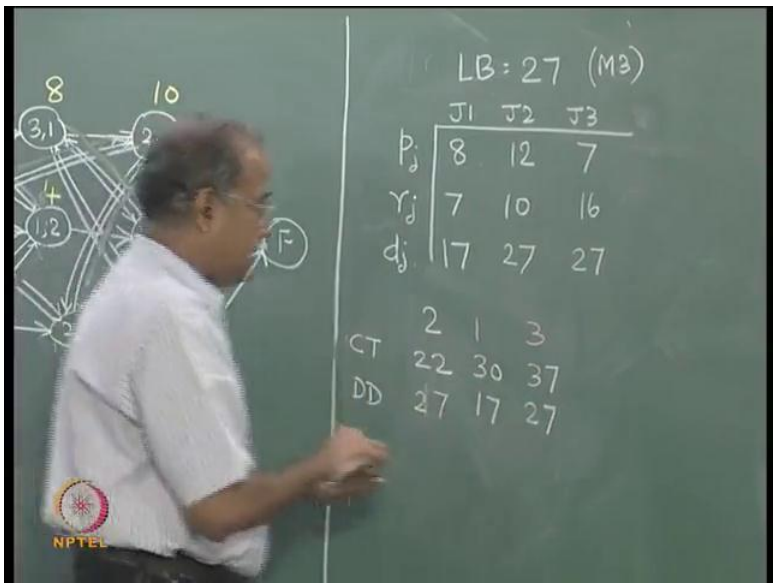
So, if the Makespan has to be 27, then the due date for M 3 has to be 17, because it requires another 10 unit of time at least, so the maximum you can stretch the due date will be 17 for J 1. So, now, our data is complete when we have to do the 1 r j L max on this, so let us do show the computation of 1 r j L max considering these 3 jobs on M 3.

(Refer Slide Time: 48:56)



So, we again start the branch and bound tree by saying, job 1 is the first one job 2 is the first one job 3 is the first job, so let us try and find out the L max for this.

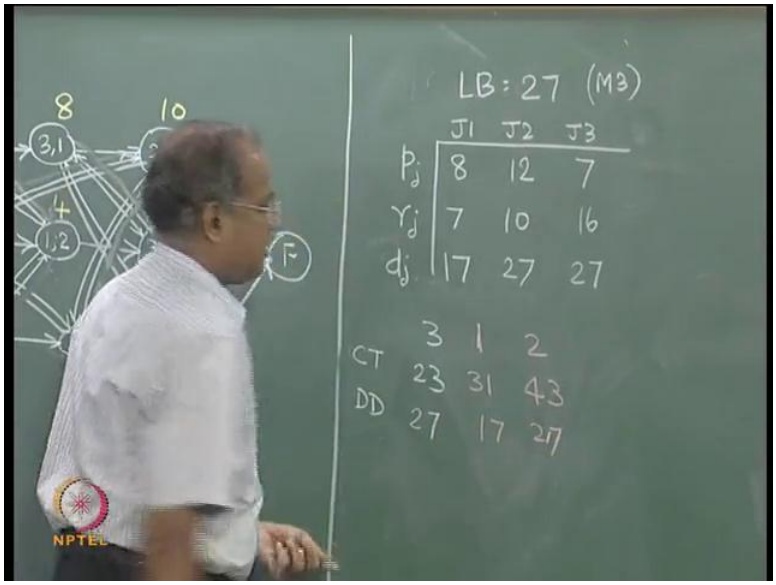
(Refer Slide Time: 49:13)



So, this will be 1 x x, but based on due dates it can be 1 2 3 or it can be 1 3 2, so let us take the case 1 2 3. Now, completion times, now this is available on 10 at time equal to 7, so 7 plus 8, 15. Now at 15 J 2 is available, 15 plus 12, 27, and at 27 J 3 is available 27 plus 7, 34. Now, due date is 17 here, for 2 also 27, for 3 also 27, there is only one l, so L max is 7, so lower bound is 7. In fact, if we have done 1 3 and 2 we actually realize that

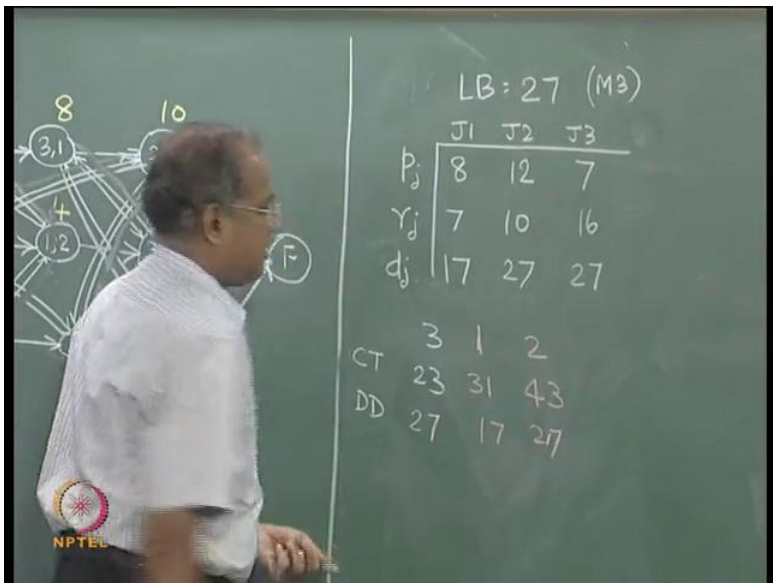
this 15, but at 15 this is not available, so L max would go up only, so this is better than the other one, since we are using a lower bound let us look at 7 as the lower bound.

(Refer Slide Time: 50:42)



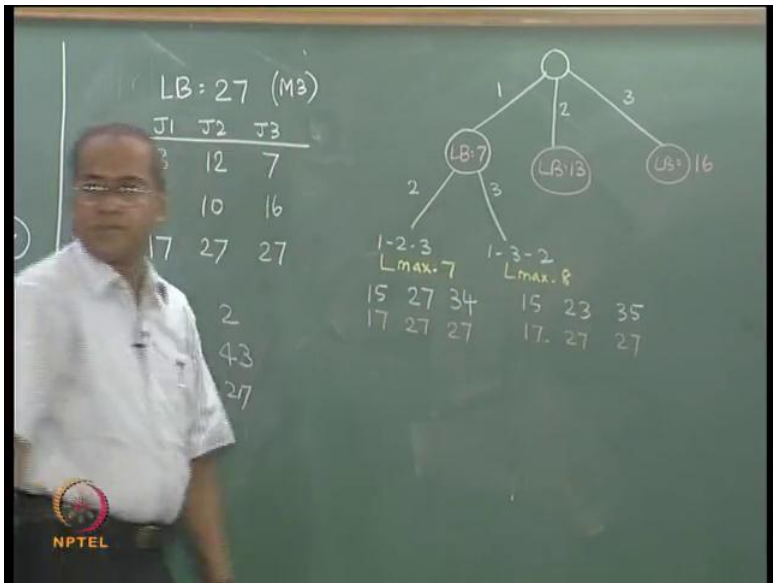
Now, we look at 2 as the first job, so 2 r j is 10, completion time is 22 then based on due dates it is 1 and 3, so at 22 it is available, 22 plus 8, 30, 30 plus 7, 37. Now, due dates are 17, 27, 17 and 27, so this is 10, this is 13, so lower bound is 13.

(Refer Slide Time: 51:29)



Now, let us look at 3 1 and 2 based on due dates 1 and 2, now completion time this is available at 16, so 23, so at 23 this is available 23 plus 8, 31 plus 12, 43. Due dates are 27, 17, 27, so this is 16, now L max is 16.

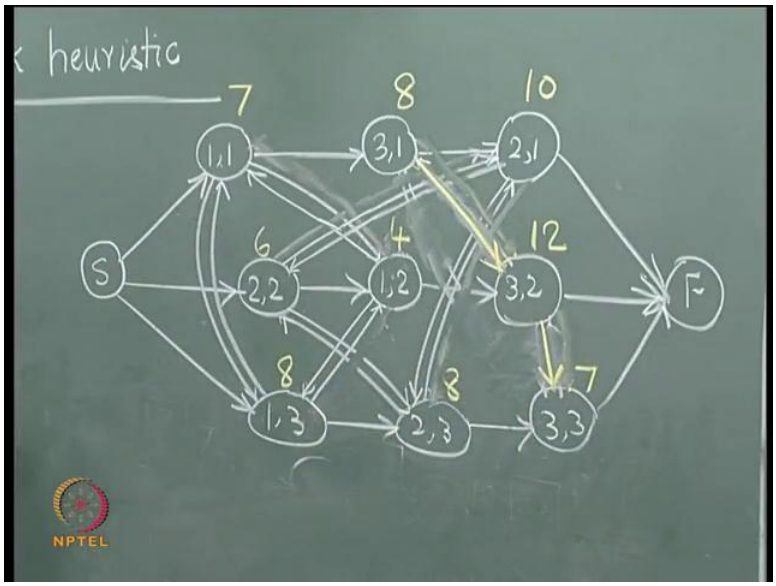
(Refer Slide Time: 52:04)



So, lower bound is 16, so we have to proceed from here, so 2 as the first, 3 as the first, so it gives us a sequence 1 2 3, 1 3 2 as the 2 possible sequences, so we will quickly evaluate the L max for each of these. So, for 1 2 and 3 completion times are 7 plus 8, 15, at 15 this is available 15 plus 12, 27, 34 due dates are 16, 27, 27, so L max is equal to 7. Now, for 1 3 and 2, so one is available at 7, 7 plus 8 15, this is available at 16, so 16 plus 7, 23, and then at 23 is available 23 plus 12, 35, due dates are 16, 27, 27, so L max is equal to 8.

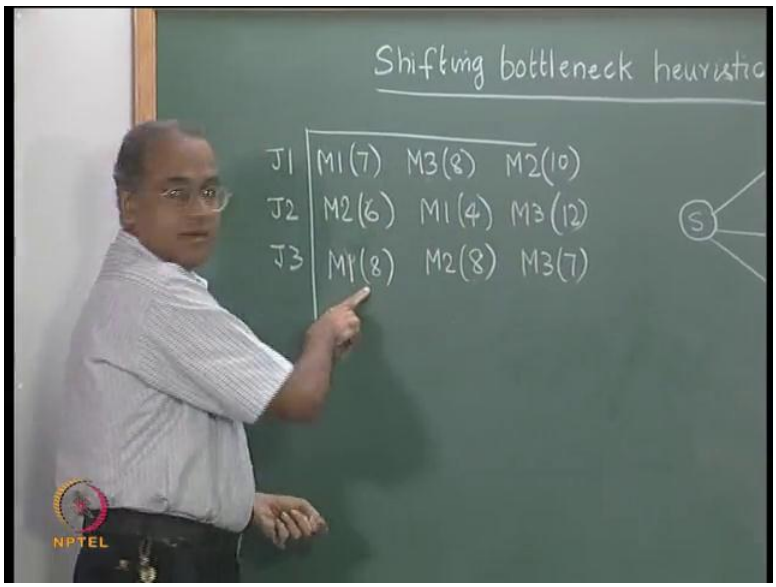
So, with L max is equal to 7, actually we could have fathomed it, because with L max equal to 7, we can fathom this, we can fathom this. And since, it is a branch and bound algorithm, the value that you can get here will be greater than or equal to 7. And since, we have a solution with 7 we need not have evaluated this also, but we know that L max equal to 7 is optimal for this. So, now, we could go and say, that if we look at M 3 based on 1 r j L max, we realize that J 1 J 2 J 3 is the optimum sequence using the branch and bound algorithm.

(Refer Slide Time: 54:09)



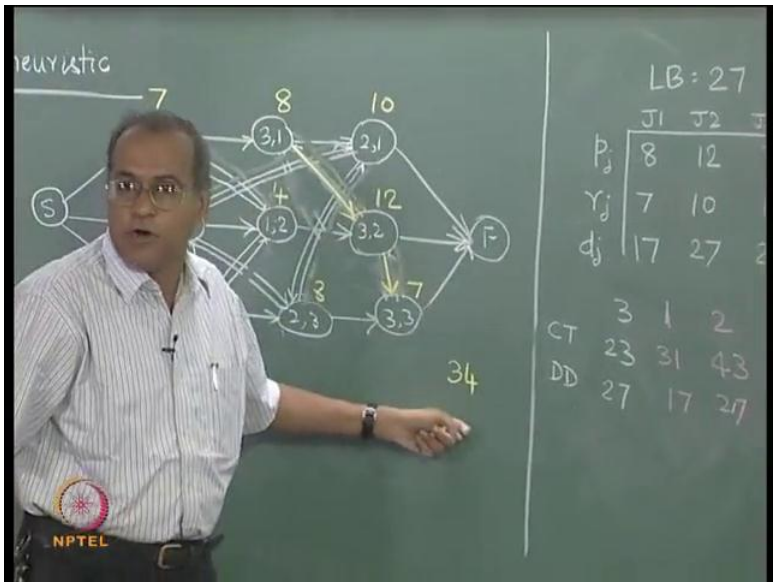
And therefore, now we can go back to M 3, machine 3 is here, machine 3 is here and machine 3 is here. Now, we say that we are going to do J 1, J 2 and J 3, therefore we replace these 2 sets of arcs by one arc which is this J 1 to J 2. We replace these 2 arcs with one arc here, and then we leave out these 2 arcs, because they are not used at all, so we have this. So, now, we have identified for M 3 what is the correct set of arcs, if we now go and find out the longest path on this network, the longest path on this network would be 27 plus 7 which is 34, because the 7 will be added to the longest path on this.

(Refer Slide Time: 55:17)



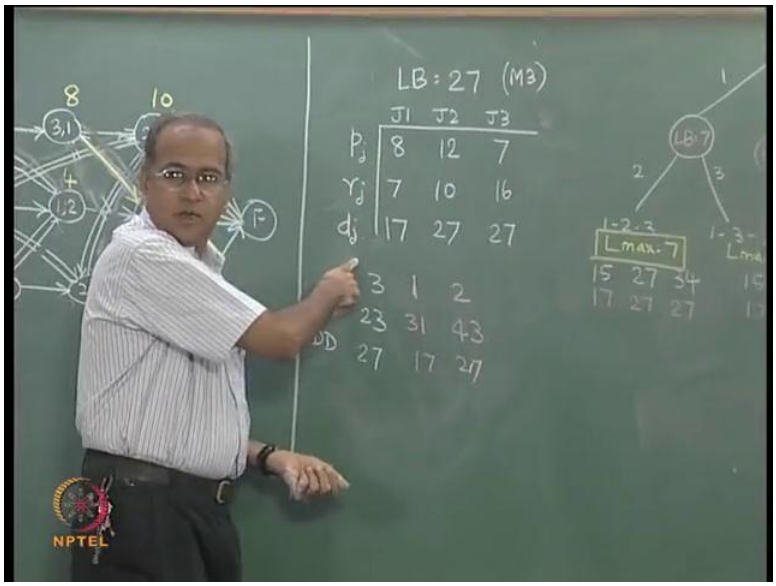
Now we have two more machines machine 1 and machine 2, machine 1 with a load of 7 plus 4, 11 plus 8, 19, and machine 2 with a load of 6 plus 10, 16 plus 8, 24. Now, machine 2 becomes the next bottleneck with 24, and then we solve a 1 r j L max on machine 2 as we did here. And then we try to a with Makespan updated to 34, because the longest path is 34, when we did for M 3 we got for 27, now we have updated the Makespan to 34.

(Refer Slide Time: 55:44)



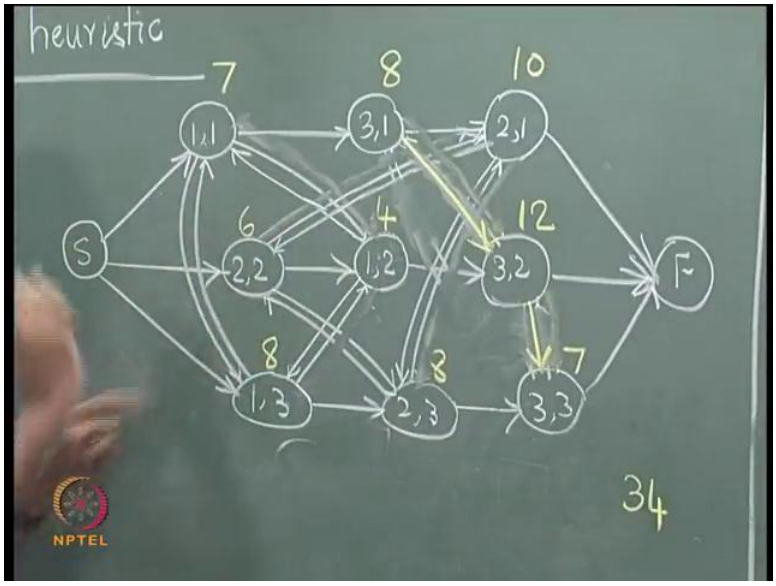
So, we solve a 1 r j L max on M 2 with Makespan equal to 34, and then we suitably define the r j's and the d j's for M 2 and so on.

(Refer Slide Time: 56:03)



Then the Makespan can increase, may increase if, so keep that and then solve it again for M 1 to get the best route for M 1. So, in this we will get the best route of jobs for M 2, as well as the best route of jobs for M 1 which means we would have chosen the best out of these set as well as the best out of the other set.

(Refer Slide Time: 56:27)



And once, we complete for all the machines, we can now find out the longest path, which will be a feasible solution, which will give us the Makespan. Now, the computations on a

machines M 2 and M 1 and application of this branch and bound idea on them. To further continue on this problem, we will see in the next lecture.