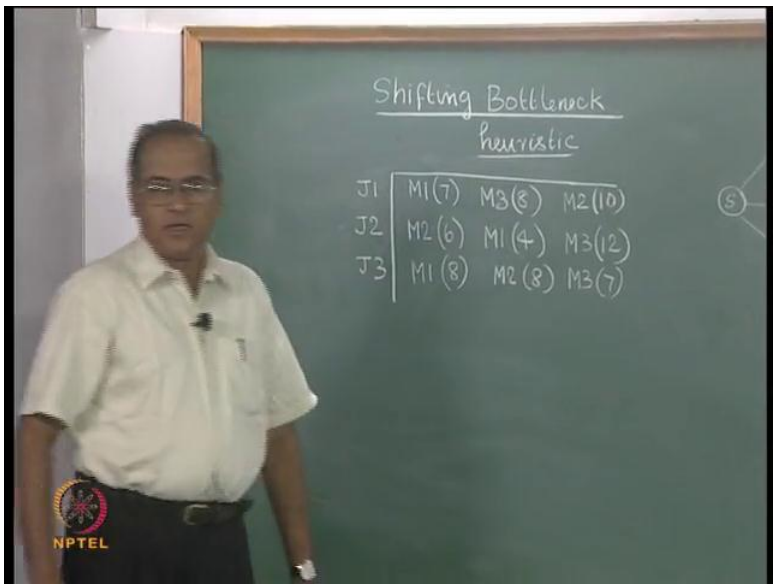
Operations and Supply Chain Management
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology Madras

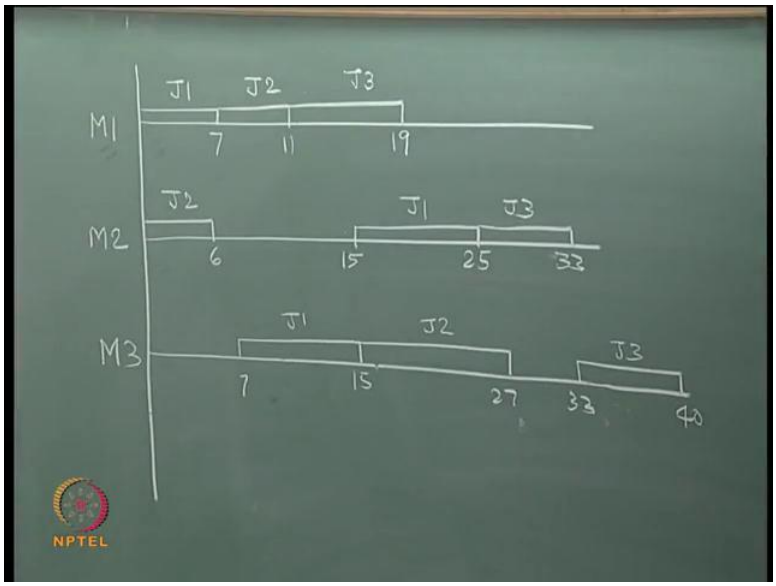
Lecture - 30
Job Shop Scheduling – Shifting Bottleneck Heuristic. Line Balancing

(Refer Slide Time: 00:15)



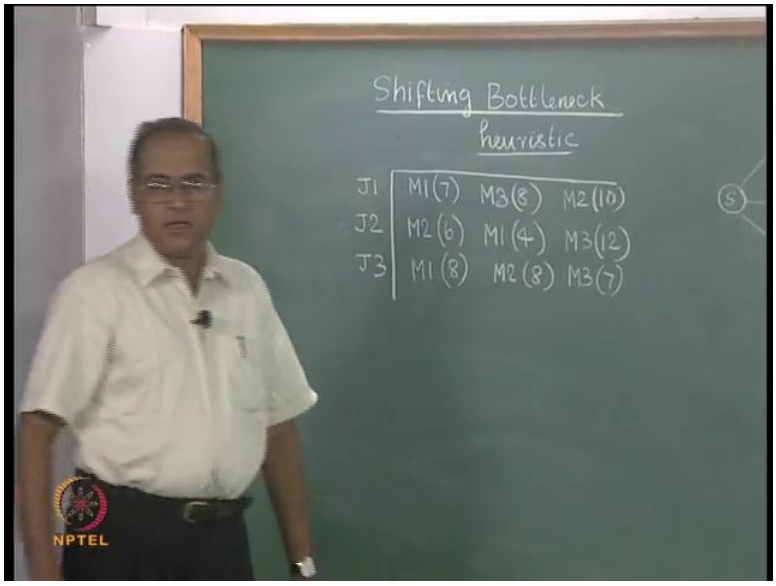
We continue the discussion on the Shifting Bottleneck Heuristic. Let me begin with a quick recap of what we saw in the earlier lecture, so we are trying to solve the problem of minimizing Makespan in a job shop, we have already seen dispatching rule based solutions.

(Refer Slide Time: 00:32)



For example, we have seen this Gantt chart, which has been prepared with shortest processing time as the dispatching rule, so if we use shortest processing time as the dispatching rule for this problem instance, we get a Makespan of 40. We are now trying to see another approach, which helps us to solve the Makespan minimization problem on the job shop, we also said that in a job shop as given in the example problem, each job has a definite route, so J 1 will go through M 1 M 3 M 2 and so on.

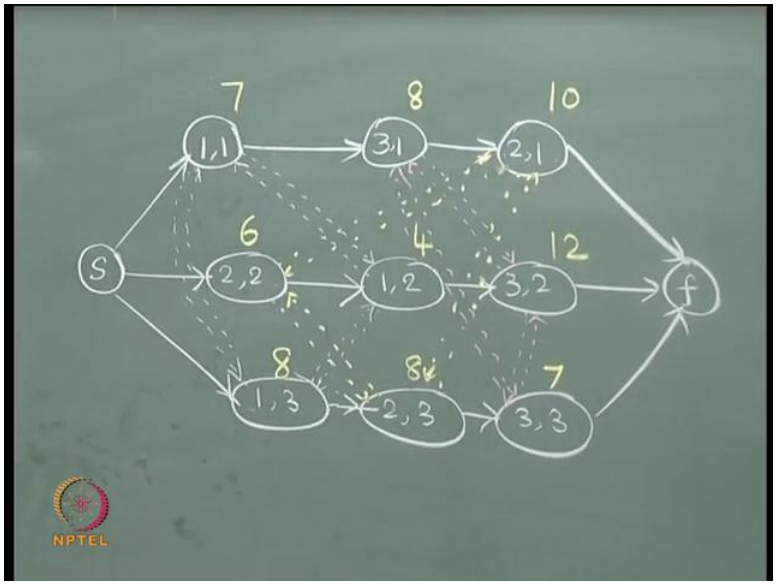
(Refer Slide Time: 01:06)



In this example all the jobs visit all the machines, so a feasible schedule will require that each machine, now treats the job visit in a certain order. For example, if we go back to the SPT solution, in this solution the jobs visit M 1 in the order J 1 J 2 J 3, M 2 in the order J 2 J 1 J 3, and M 3 in the order J 1 J 2 J 3. So, one other way of describing a schedule is to determine the order in which the jobs are going to visit the machines, the in any case if we see here job 1 has to visit all the 3 machines, but we are interested in taking each machine and the order in which the jobs come in.

That is the variable part, which we are actually try into optimize, if you are able to get the correct order, in the sense at the moment if the order on M 1 is J 1 J 2 J 3, order on M 2 is J 2 J 1 J 3, an order on M 3 is J 1 J 2 J 3 the Makespan is 40. So, the question can be posed differently as what is the order such that this Makespan is minimized, also we have to ensure that this order is satisfied, that J 1 will first visit M 1, and then M 3, and then M 2, which has been satisfied by the Gantt chart schedule. We also now have a network representation, which we have seen now the definite orders for the jobs, for example, J 1 will visit M 1 M 3 M 2 is Shown here as J 1.

(Refer Slide Time: 02:59)



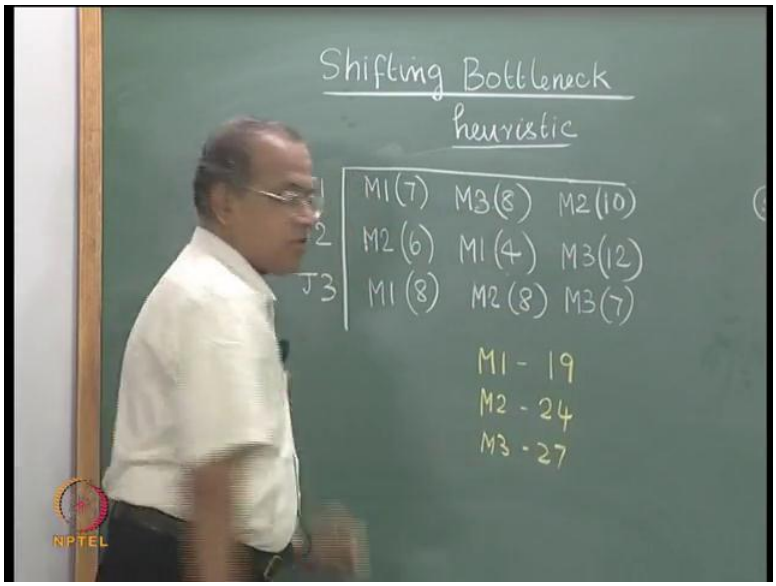
Here, i j j represents the job, and i represents the machine, so J 1 first visits M 1 then M 3 and then M 2 and finishes, so this part has been shown through these fixed arrows in this network and through dark lines here. What are shown through dotted lines is the decision that we have to make, for example since a machine can process only one job at a time.

So, if we take machine 1, then machine 1 figures here for J 1, here for J 2, and here for J 3, so what we do is we draw 2 arcs, which are dotted, joining 1 to this in this direction and the other in the other direction.

So, for every pair we have that, so since we have 3 jobs, we actually have 1 to 2, 1 to 3 and 2 to 3 each has 2 arcs. So, we have 6 dotted arcs, the ones related to machine 2 are shown in yellow, and the ones related to machine 3 are shown in a kind of a dark brownish pink color. So, we now have to find out, if we take out of if we take machine 1 alone, out of these 6 arcs, we actually find in the end only 2 arcs will be active, which 2 are active is what we have to find out.

If we want to map the SPT sequence on this, and for example if we take M 2, then the order is J 1 J 2 J 1 and J 3, which means on M 2 it will be 2 to 1 and 1 to three. So, on M 2 it will be first 2 to 1, which means 2 to 1 and then 1 to 3, so out of these 6 arcs which are in yellow, the 2 to 1 here 2 to 1 and then 1 to 3 will be active. Now, we wish to find out for each machine, which will be the active arcs out of the dotted arcs. So, we set out to do that, and then we also related this aspect to solving a 1 r j L max problem on the bottleneck machine, we also identified first M 3 as the first bottleneck machine by first calculating the loads on each of these machines.

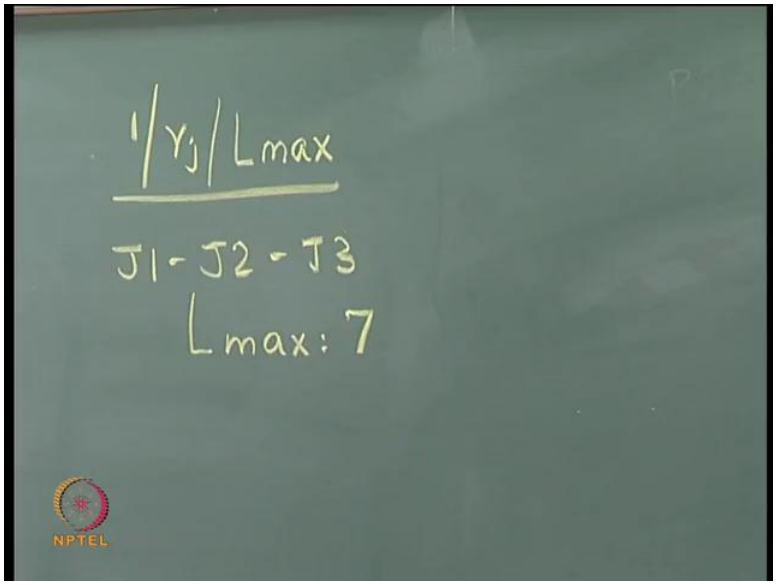
(Refer Slide Time: 05:43)



So, M 1 has a load of 7 plus 4, 11 plus 8 19, M 2 has a load of 10 plus 6, 16 plus 8, 24, and M 3 has a load of 8 plus 12, 20 plus 7, 27. So, we took M 3 as the first bottleneck

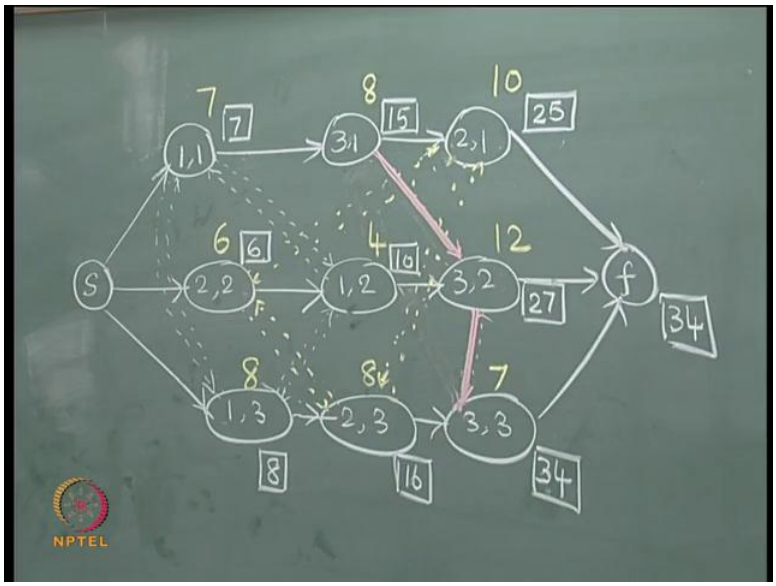
machine, and then we solved a 1 r j L max problem, which means problem of minimizing the maximum tardiness on a single machine with release times.

(Refer Slide Time: 06:23)



So, 1 r j L max on M 3, so we solved a 1 r j L max problem, where one represents single processor, r j means released times for job j, and L max means maximum tardiness that we wish to minimize. So, we solved this problem on M 3 and then we got the sequence J 1 J 2 J 3 which minimized this, so we got the sequence J 1 J 2 J 3 on M 3 with L max equal to 7. Now, we have to put the solution or superimpose the solution on this network, so we have solved for the third machine, so we have solved for this set of 6 arcs. And then we have J 1 J 2 J 3 coming in, which means on M 3 we will first do J 1, and then we will do J 2, and then we will do J 3.

(Refer Slide Time: 07:35)



So, on M 3 we will first do J 1, so this is 3 1 would mean job 1 on machine 3, first we do this then we do from this, so this arc becomes active, and then from here we will do this, so this arc becomes active. The rest of the dotted pink arcs are now removed, because they are no longer required, so we will remove this we would also remove all this, so we just have these 2 arcs coming in to the picture.

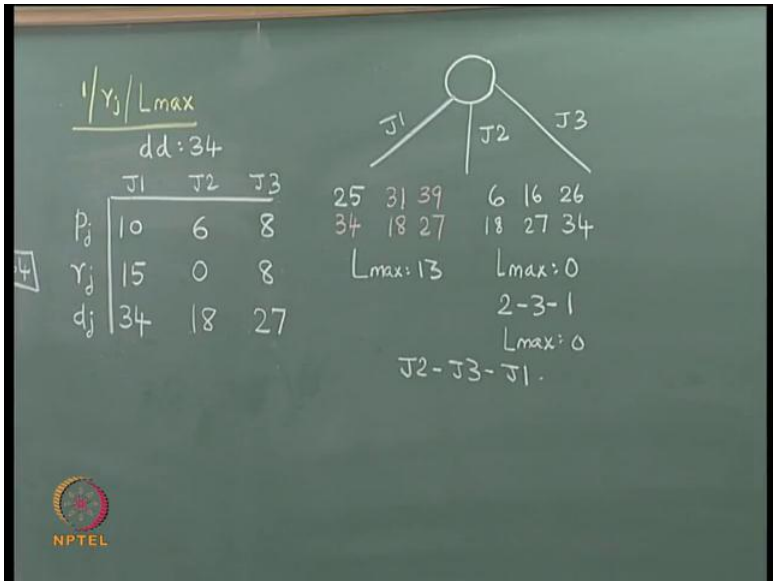
Now, if we find the let us mark it again with the thicker line, now the longest path of the on the network gives once again a lower bound to the Makespan. Earlier the longest path on this network was the sum of the job processing times 7 plus 8 15 plus 10, 25, 6 plus 4 10 plus 12. 22. 8 plus 8, 16 plus 7, 23. But now with the inclusion of these 2 arcs, if we compute the longest path on this network would become 34. So let us just show the computation of a longest path on this network.

So, let us say we start with 0 for this, so this comes at 7 it depends on how we write it, because it is activity on node network, so we could say that this one finishes at 7 from here. So, this would mean 7 plus 8, 15, 15 plus 10, 25, now what we do is this is 6, this is 6 plus 4, 10. Now, this one will be 15 plus 12, 27. 10 plus 12, 22, so 27, now this will be 8, 8 plus 8, 16, now this will be 16 plus 7, 23, 27 plus 7, 34 and the finish will be 34.

Now, the lower bound on the Makespan increases and it becomes 34 - 27 plus 7 more importantly it is the longest path on the network. Now, we look at the next bottleneck,

which is M 2 and then we try and solve the 1 r j L max problem on M 2, on M 2 with 34 which is the lower bound of the Makespan, so that is updated as the final due date.

(Refer Slide Time: 10:48)



So, with due date equal to 34 we solve 1 r j L max on M 2, so we have J 1 J 2 and J 3, and we have the processing times p j, p j on M 2. So, J 1 on M 2 is 10, J 2 on M 2 is 6, and J 3 on M 2 is 8, then we write r j and d j, r j is the release time at which it is available, and d j is the due date. Now, if we look at M 2 and we look at J 1, so J 1 on M 2 is the third operation, so earliest J 1 is available on M 2 is 7 plus 8 15.

So, 15 is the time earliest that J 1 is available on M 2, because at J 1 has to finish processing on M 1 and M 3, and the earliest it can be available is 7 plus 8, so r j is 15 here. Now, as far as J 2 is concerned J 2 on M 2 is the first operation. Therefore the earliest it is available is 0, as far as J 3 is concerned M 2 is the second machine, and therefore the earliest J 3 can be available at M 2 is at time equal to 8.

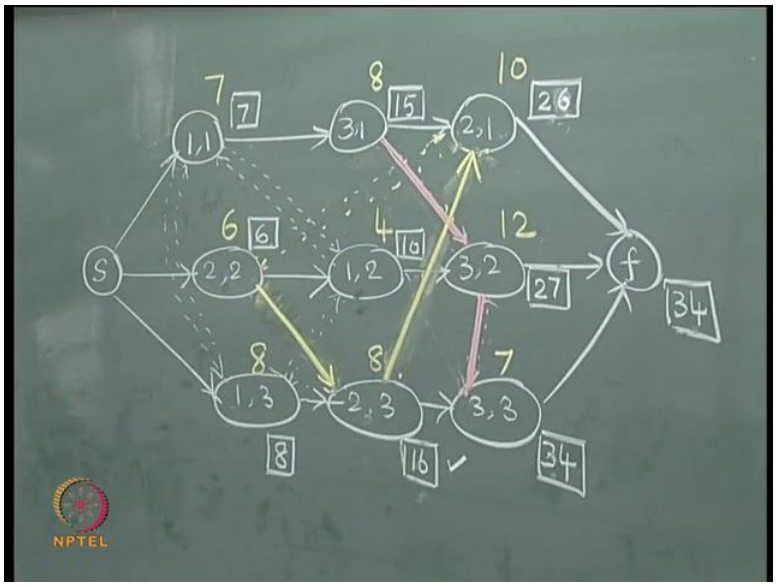
Now, as far as the due dates are concerned J 1 on M 2 is the last operation, so it is enough if it is available at time equal to 34. As far as J 2 is concerned J 2 on M 2 is the first operation. So, the latest due date the farthest, we can think of is 12 plus 4, 16 it definitely requires at least 16 more units to finish, so the due date can be 34 minus 16 which is 18. And as far as J 3 is concerned it is once again the last operation, J 3 on M 2 is last, but one operation. So, at least 7 more units are required, therefore the due date can be 34 minus 7 which is 27. So now, we have to solve a 1 r j L max on M 2.

So, we start with the starting node and then we put J 1 as the first job, we start another sequence with J 2, and we start another sequence with J 3 as the first job, so when we do this let us start writing the completion times. Now, when we put J 1 as the first job, J 1 is available only at time equal to 15, so a completion time is 25 for J 1, and then by the preemptive EDD rule, we would based on the due dates we have J 2 and then J 3. Now, J 2 is available at time equal to 0, so J 2 can finish at 31, because it is available at 25 plus another 6 is 31 by 31 J 3 is also available, which is available at 8.

So, 31 plus 8 39, due dates are 34 18 and 27, so here L max is 13 here L max is 12, so lower bound on L max is equal to 13. Now, when we start with J 2, so J 2 is available at time equal to 0, so it will finish at 6, and then J 3, and then J 1 by the EDD rule, now this is available at 8 this is available only at 15. So, you have to definitely wait till 8, so 8 plus 8 16, now this is available at 15 16 plus 10 26, now the due dates are 18 27 and 34, so lower bound on L max is equal to 0, because all of them are within the due dates.

And we also realize that actually what we have done is we have evaluated a feasible solution, with 2 3 and 1 with L max equal to 0, this is the feasible solution with J 2 first, then J 3 and then J 1 with L max equal to 0. Therefore, we can stop the algorithm here itself, for a maximum tardiness problem if you have a solution with maximum tardiness equal to 0, a feasible solution with 0, then it is optimal. So, we could stop right here, and say that the sequence on J 2 on M 2 is J 2, J 3, and J 1, now we superimpose this on M 2.

(Refer Slide Time: 16:59)

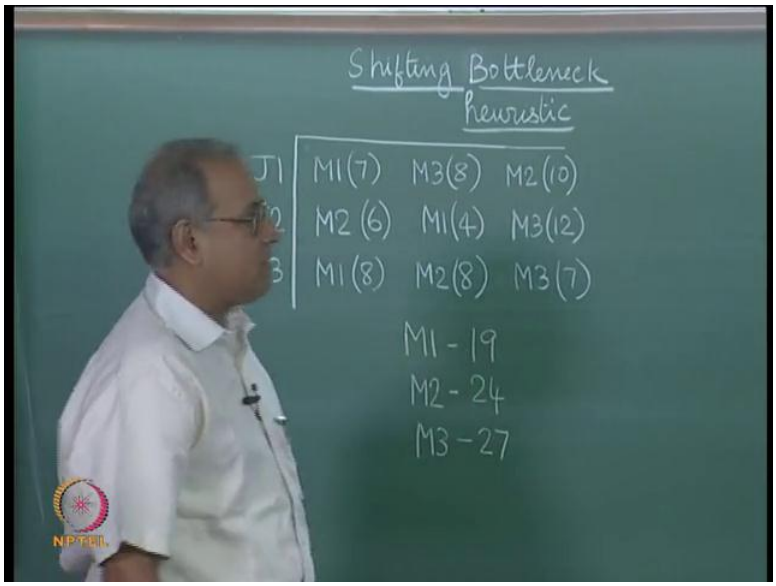


So, M 2 is shown with the yellow color, so these 6 arcs are there, so the sequences J 2, J 3, and J 1, so first on M 2, we will first do J 2, then we will do J 3 and then we will do J 1. So, we have to update this, so first we do J 2, so we do this first, then we do J 3, so we come to 3, and after that we go to 1 through this, so this is the sequence that will happen on machine M 2.

Now, with this sequence we now need to update the lower bound, so the longest path on this network will give us an updated value of the lower bound, so that; obviously, the lower bound now can only be 34 or more, because we are adding arcs on this network. Therefore, the longest path cannot decrease, so now, we have to once again find what are these node labels, so that we find out the lower bound on the Makespan. So this is at 7 this is again at 15, so this is at 27, now this is at 8, now this one is 6 plus 8, 14, but then we have 8 plus 8, 16, so this stays.

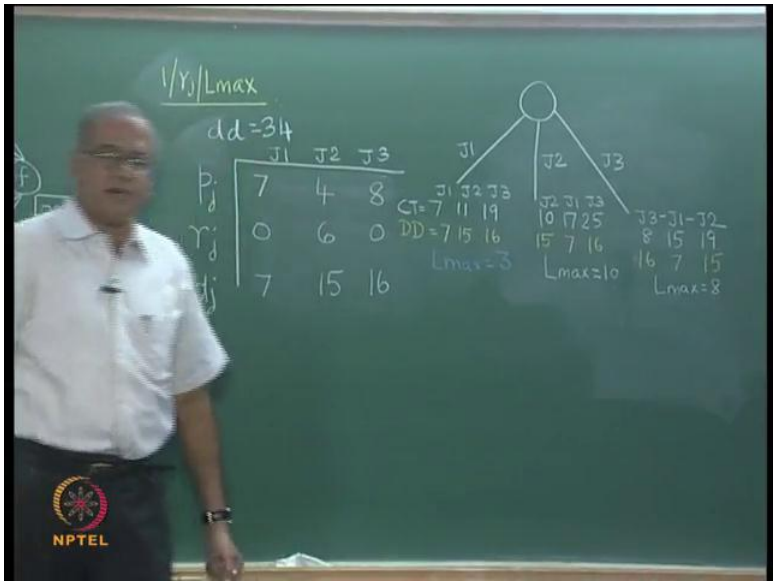
Now, this will become 16 plus 8, 16 plus 10, 26 whereas, earlier it was 15 plus 10, 25, so this will become 26, otherwise 15 plus 12, 27, 27 plus 7, 34, 16 plus 7, 23, 34, so the lower bound remains at 34 with this. The only change in the label is the change here, that this is 16 plus another 10, 26, whereas here it was 15 plus 10 25 earlier, so the lower bound does not change with the inclusion of these 2 arcs, but only one of the node labels changes. So, now, we look at the third machine which is M 1, we have already seen M 3 and M 2, so we now need to do a 1 r j L max on M 1. So, we do a 1 r j L max on M 1 with the same due date as 34, because the longest path in this network right now is 34.

(Refer Slide Time: 20:05)



We now solve the 1 r j L max problem on the third machine, which is machine M 1, we have already solved the 1 r j L max problems for M 3 and then M 2. So, now from this network, we observe that the processing times, for the 3 jobs J 1 J 2 J 3 on M 1 are 7 for J 2 it is 4 and for J 3 it is 8.

(Refer Slide Time: 20:29)



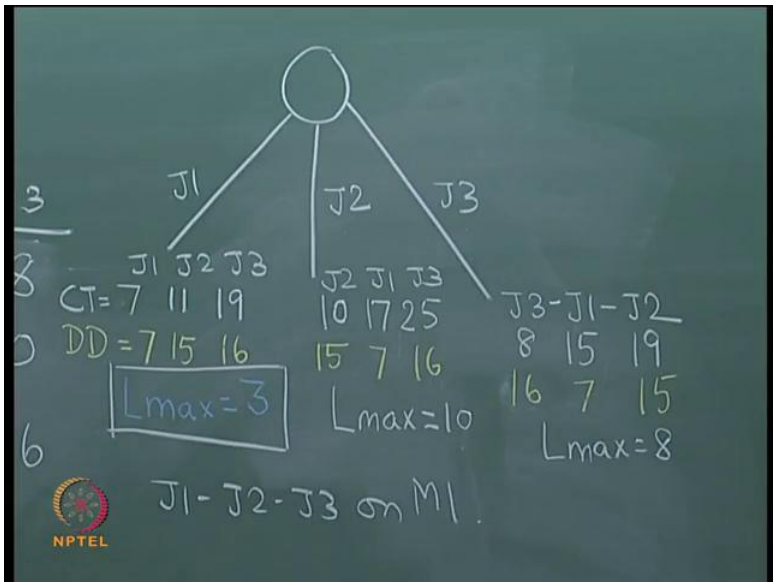
Now, we have to compute the released times r j as well as the due dates d j, so if we take job J 1 on M 1, now the earliest this is available is time equal to 0, so r j will be 0 for this. Now, here the earliest this is made available is after it goes though this machine,

therefore r j will be 6 and as far as J 3 on M 1 is concerned the earliest it is available is at time equal to 0. Now, we look at the due dates, the due date remains as 34, now when we look at this the longest path that connects will be 7 plus 8, 15 plus 10, 27, so 34, so 34 minus 7 is 27, 27 minus 12 is 15, 15 minus 8 is 7, so due date for this will have to be 7, because the longest path is 7 plus 8 15 plus 12 27 plus 7 34.

Now, as far as this is concerned the longest path would be from here, so 34 minus 7, 27, 27 minus 12 will be 15 for this to come, so due date will be 15. And for J 3 which is here, the 34 minus 10 plus 8 would give us 17, 34 minus 15 would give us 19, so this will become 17. So let me compute this again. Now, as far as J 3 on M 1 is concerned this is the one, so 7 minus 8, 15 would give us 19 from this side it will be 10 minus 8, 18 therefore, at time equal to 16 this should be available, so that 16 plus 18 becomes 34.

So, this is not 17 this becomes 16. So let me again explain the computation of the due dates, J 1 on M 1 is here and the longest path is 7 plus 8 15 plus 12, 27 plus 7 34, which gives us this 34. So, in order to meet this head line of 34 this should be over by time equal to 7 plus 12, 19 plus 8, 27, so 34 minus the remaining processing time of 27 gives us a due date of 7. Now, as far as this is concerned once again in the due date earliest or latest that this has to be completed comes from 12 plus 7, it has remaining 19 time units of processing, and since it has to finish at 34 the due date will be 15. Now, as far as J 3 is concerned we are here, so again we go through the path, so 34 is the final at the moment the Makespan is 34, so 10 plus 8, 18 more time units of processing is required, so the due date becomes 16 for this.

(Refer Slide Time: 24:47)



Now, with this data we now try and solve a 1 r j L max problem on machine 1, so we now begin with J 1 as the first job in the sequence, J 2 as the second job first job, and J 3 as the first job in the sequence. So, when we look at J 1 as the first job in the sequence, now it is available at time equal to 0, therefore the completion time will be 7, following the earliest due dates sequence we try to get J 2 inside. So, J 2 is available at 6, now the machine is available at 7, so this will become 7 plus 4, 11, by which time J 3 is available, so plus 8 19, so these are the completion times, the due dates are 7, 15 and 16, so only one job is tardy.

So, from this we find that L max is 3, we also observe that we get a feasible solution, when we actually apply this preemptive EDD rule, we get a feasible solution with L max equal to 3. Now, with J 2 as the first job in the sequence, so J 2 starts at 6 completion time is 10, now going by the preemptive due date J 1 comes next, so J 1 starts at 10 and finishes at 17 by which time J 3 is available. So, J 3 will take 17 plus 8, 25, so now, the due dates are 7, 15 for the for job 2 the due date is 15, for job 1 the due date is 7, and for job 3 the due date is 16 So, this gives us a solution with L max equal to 9.

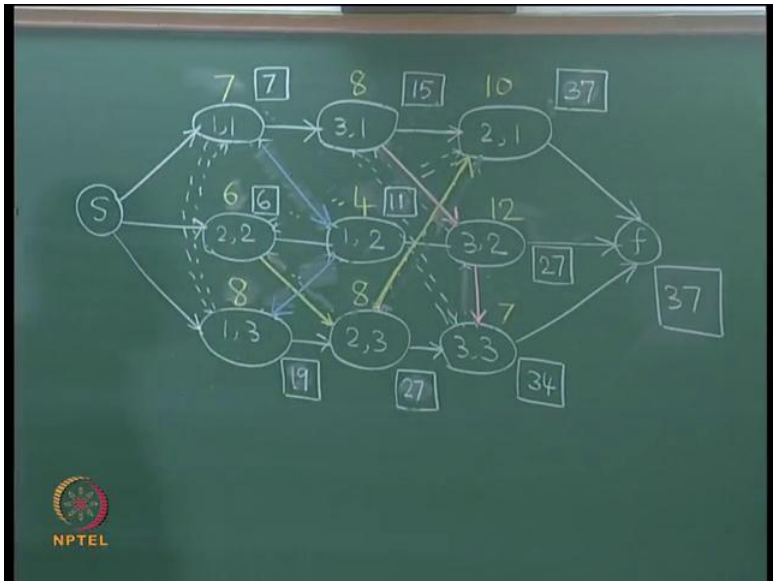
Now, this job this is the sequence this is J 2 J 1 J 3, this is J 1 J 2 J 3, so the sequence J 2 J 1 J 3 will give us a solution with L max equal to 9. Now, this job is early this job is tardy with due date equal to 7 and completion time equal to 17, so L max is equal to 10 from this, and for J 3 due date is 16 completion time is 25, so tardiness is 9, so maximum

tardiness is 10. Now, if we consider the third one with J 3 as the first job, preemptive EDD would give us the order J 3 followed by the lowest one J 1 and then J 2.

So, we get J 1 J 2, so here the completion times will be starts at 0 and finishes at 8, J 1 is available at time equal to 0, so finishes at 15 by which time J 2 is available. So this finishes at 19. Now, due dates are 16 for J 3, 7 for J 1, and 15 for J 2, now we realize that J 3 is ahead of the due date this is behind the due date by 8 units, this is behind by 6 units, so we get L max equal to 8. Now, when in these 3 nodes what we have done is we have started with J 1 J 2 and J 3, and we are able to get feasible solutions by applying the preemptive earliest due date rule.

We may also a compute the lower bound and proceed, but finally, when we solve this problem, we will get this solution with minimum L max equal to 3. So, 1 r j L max problem talks about the sequence J 1, J 2, J 3 on M 1 with L max equal to 3, now we go back and try to map this on this network, so on M 1 the sequence is J 1 J 2 J 3.

(Refer Slide Time: 30:04)

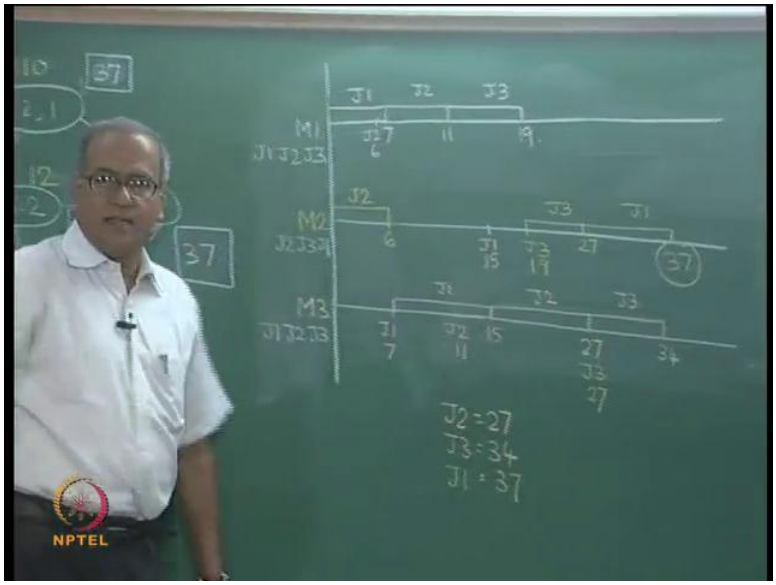


So, J 1 first then J 2 and then J 3, so we map this, so J 1 to J 2 will come here, and then we will do J 2 to J 3, which will now come here, so now, this is the final network, where on M 1 J 1 followed by J 2 followed by J 3. Now, we have to compute the longest path, and update it now we do the computation of the node labels, so we have 7 plus 8, 15 and this one will become 8 plus 8, 16 plus 10, 26. Now, this will become 7 plus 4, 11 instead of 6 plus 4, 10, so this will become 7 plus 4, 11, now this node will become 11, 11 plus 8

19 this will become 19, so this will become 19 plus 8, 27, 27 plus 7, 34, but 8, 27 plus 10 will become 37 here.

Now, this one will become 11 plus 12, 23, 15 plus 12, 27, so finally the longest duration will be the maximum of 37, 27 and 34, and we will get a solution with 37. Now, the Makespan is 37 and we have a feasible solution, which Makespan equal to 37. You may also observe that earlier the Makespan was 34, we computed L max equal to 3, so the L max gets added and 34, finally becomes 37 which is the Makespan for the sequence obtained using the shifting bottleneck heuristic. Now, one way of representing it is on this network, the other way of representing it is on the familiar Gantt chart, and now we will represent this schedule on the familiar Gantt chart, which we are used to.

(Refer Slide Time: 32:46)



So, we will, now represent it on the Gantt chart, this will be M 1, this is M 2, and this is M 3, so the sequence on M 1 from this comes from 1 1 1 2 and 1 3, so the sequence is J 1 J 2 J 3. Now, on M 2 we realize that this is first, so J 2 J 3 and then J 1, so J 2 J 3 and J 1 and on M 3 it is again 1 2 and 3, so J 1 J 2 and J 3. Now, we have to map this on the Gantt chart, so we look at J 1, J 1 first goes for processing on M 1, so J 1 on M 1 finishes at 7, and at 7, J 1 goes to M 3.

Now, J 1 is actually the first job on M 3, so J 1 has come at time equal to 7, and finish at 15, and at 15, J 1 will go to M 2. J 1 comes here at 15. J 1 is the third job to be processed on M 2, so we will just leave J 1 here, and we will take up J 1 after J 2 and J 3 have been

processed. Now, we look at this J 2's first operation is on M 2, so J 2 will start at time equal to 0 and finish at 6, and at 6 J 2 will come to M 1.

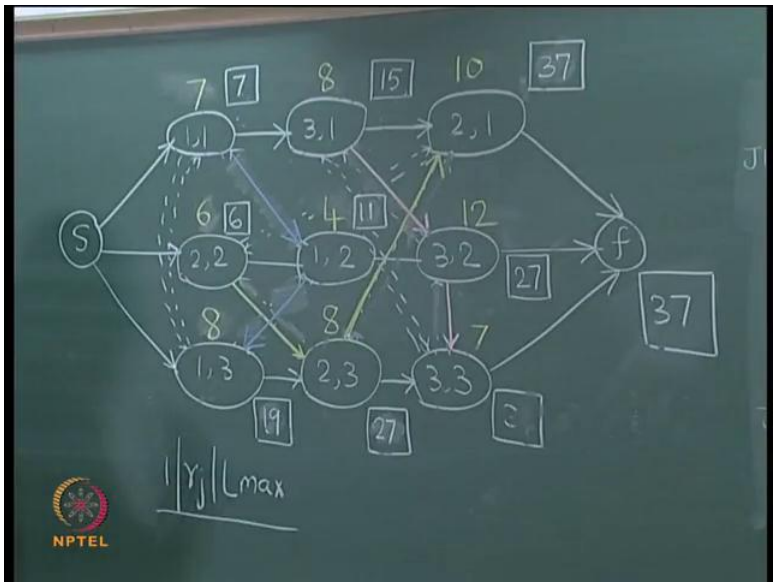
Now, J 2 is the second job to be done on M 1, and J 1 is the first J 1 is already over J 2 has come at 6, so it will take up J 2, so J 2 on M 1 will be 7 plus 4, 11, and J 2 will now move at time equal to 11 to M 3, so J 2 will come here J 2 will come at 11. Now, as far as M 3 is concerned J 1 is already over J 2 is the second job, so J 2 is ready, so J 2 on M 3 J 2 starts at 15 takes 12 units of time from here, and therefore 15 plus 12, 27. J 2 will finish and J 2 completes all the 3 activities at time equal to 27. Now, let us look at J 3. J 3's first operation is one M 1, so J 3 will be taken up at time equal to 11 and it will go from 11 to 19.

So, J 3 will finish at 19, and J 3 will come to M 2 at time equal to 19, now J 3 is the second job J 2 is already done, so J 3 will start at 19, so J 3 on M 2 will be 19 plus 8, 27. So J 3 will finish at 27, and at 27, J 3 will now go to M 3. So J 3 will come here at 27. Now, J 1 J 2 are already completed J 3 is the only one available, so J 3 on M 3 is another 7, so J 3 will start at 27 and J 3 will finish at 34, so J 3 will finish everything at time equal to 34.

Now, we look J 1. J 1 has actually come here at time equal to 15, so J 1 on M 2 is another 10, so this can start at 27 only takes 10 more time units and finishes at 37. So J 1 will finish at 37, which is indeed the Makespan. And this 37 is the same as this 37 that we have computed, so we can do either of these, now from the shifting bottleneck solution, we can map it on to the Gantt chart like we did. If we get a Gantt chart solution, it is also a possible to map it on to this network and the longest path on this network will be the same as the longest of the bars on the Gantt chart and for this problem the Makespan is 37.

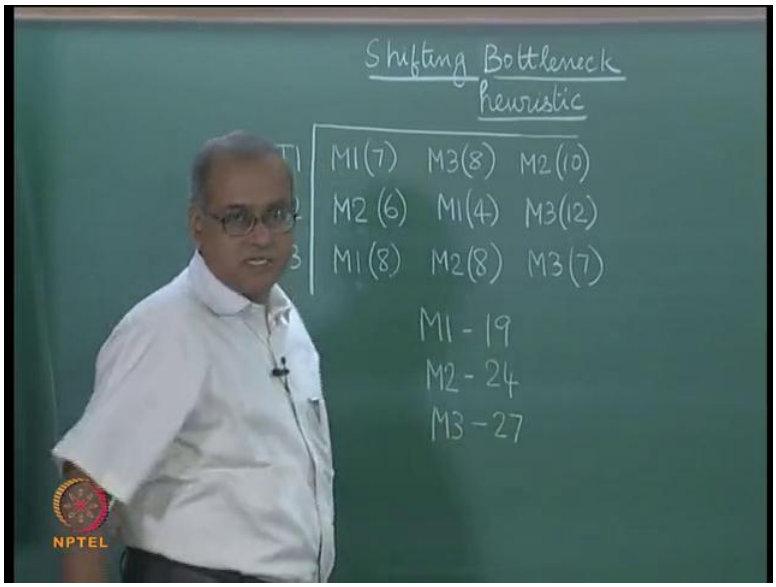
Now, we have to check whether this Makespan is optimal or does the shifting bottleneck heuristic guarantee optimality, the answer is the shifting bottleneck heuristic does not guarantee optimality. What it tries to do is either when we look at it through this network or through the Gantt chart, it takes one machine at a time and tries to find an optimum sequence not for the Makespan problem, but for the 1 r j L max problem.

(Refer Slide Time: 39:16)



And then it adds on this L max into the existing Makespan, and updates the existing Makespan. So, on the one hand it solves an optimization problem for a particular machine, but the order in which the machines are taken will is now based on the loads.

(Refer Slide Time: 39:46)



Say here, we took the machines in the order M 3 followed by M 2 followed by M 1, this order need not be the optimal order, therefore the shifting bottleneck heuristic does not guarantee the optimum solution. Even though, it solves an optimization problem at each stage, we also have to observe that 1 r j L max problem is actually a difficult problem to

solve, but luckily the branch and bound that we used is able to use solves problems of reasonable number of jobs optimally.

Therefore, we are convinced that we can use the 1 r j L max on a single machine and apply it sequentially on the machines. The thumb rule that we used here we first did it on M 3, then on M 2, and then on M 1, because our choice was based on the work load on each of these machines. Now, this particular order of applying the 1 r j L max need not be optimum, so shifting bottleneck heuristic does not guarantee the optimal solution, but it provides an excellent alternative to a dispatching rule based approach to solve the problem, because any dispatching rule based solution would give us a certain Gantt chart.

Now, this Gantt chart on one hand computes the Makespan on the other, also tells us the order in which the jobs are going to be processed on each machine. Now, this order can always be mapped on to the network, and the longest path would give us. Alternately the shifting bottleneck heuristic solves a 1 r j L max problem on each machine, and then tries to find out this order on each machine and maps it on to the network.

So, this is a very powerful heuristic does not guarantee optimality and is completely different from dispatching rule based approach. Even though in principle both the dispatching rule based approach as well as the shifting bottleneck heuristic, try to find out what is the best sequence of jobs on each of these machines. Now, this method became very popular largely because of it is ability to give optimum solutions in many instances, and close to optimum solutions in many instances as well.

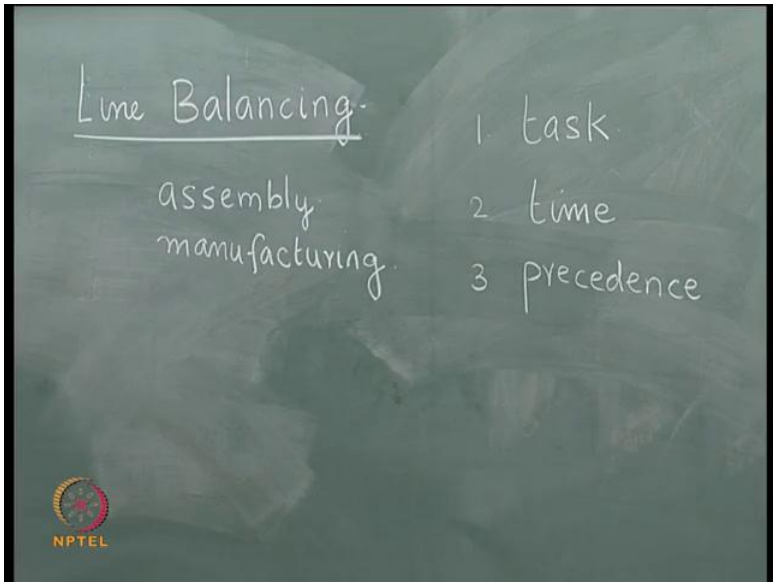
In fact, if one looks at the job shop scheduling literature there are 3 important or difficult problems, which are called the Fisher and Thompson problems and particularly the second of the Fisher and Thompson problem. For which people knew the optimum Makespan, but we are not able to get that comfortably using dispatching rules. The shifting bottleneck heuristic, for that particular problem instance gave the optimum solution.

So, the shifting bottleneck heuristic is important for many reasons, one it is able to provide very good solutions, even though not optimal on all occasions, it is able to give optimum on many instances and close to optimum on very many instances. It provides an

alternative to a dispatching rule based approach to solve a job shop scheduling problem, because this sequence 1 2 3 or 2 3 1, 1 2 3.

Suppose, we take a shortest processing time based Gantt chart, then it gives a certain sequence for each machine that is obtained through a dispatching rule. Here, the corresponding sequence is obtained by solving a 1 r j L max or by solving an optimization problem, therefore it can give better solutions than using a single dispatching rule.

(Refer Slide Time: 44:09)



So, this is used effectively used to solve job shop scheduling problems to minimize Makespan.

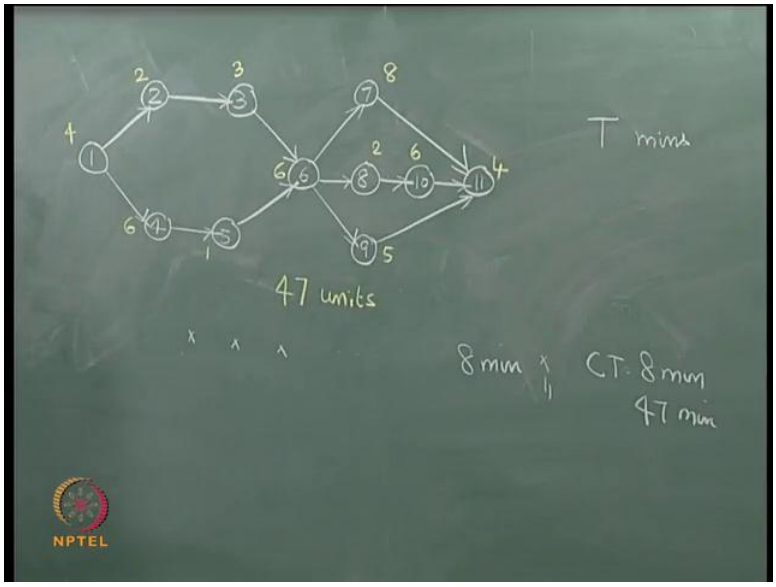
Now in the line balancing problem, essentially we use the line balancing idea in assembly, as and also these ideas can be used in manufacturing. Now, what is the assembly line balancing problem? Now let us assume that we are trying to assemble a product. Now, this assembly involves assembling or fixing several components and sub assemblies into a main assembly, so first and foremost in the assembly problem, there are components that are brought into the assembly and assembly of each one of them is now called a task.

So, an assembly would involve 10 or 12 or 15 tasks, where 15 different subassemblies and components are assembled into a final product or final assembly. Now, each task

also has a processing or a task time, so there is a time associated with each of these tasks. Now, many times if there are 10 or 12 parts and sub subassemblies that go the assembly, there has to be an order in which these have to be assembled.

Therefore, there are precedence relationships, which may say that I cannot assemble or I cannot do task 3 unless I finish task 2. So, an assembly line balancing problem the data that are required to do this are the number of tasks, the time on for each of these tasks and the precedence relationship of the assembly.

(Refer Slide Time: 47:00)



So, let me take an example to explain all these, and then try and use the same example to solve the line balancing problem. So this network shows an assembly line balancing problem with 11 tasks, which are shown as 1 to 11 as nodes of the network. Now, the task times are shown as the arcs, and these are 4, 2, 3, 6, 1, 6, 8, 2, 5, 6 and 4. Now the number shown in yellow are the time taken to do each of these tasks. And they are shown as node labels, associated with the node, the precedence is captured through the arcs. For example from this network one can say that we can perform task 2, after we have performed task 1, task 1 is a preceding activity for task 2.

Similar, task 1 should precede task 4, similarly 3 and 5 should precede 6, which means 6 cannot be done till 3 and 5 are completed 6 cannot be taken up and so on, so the precedence relationship are captured in the form of these arcs on this network. Now, without loss of generality we can assume that the order 1 2 3 4 5 6 7 8 9 10 11 is feasible,

so there is a feasible solution to this problem. For example, we will not have a situation, where one has to precede 2 two has to precede 3 and 3 has to precede one, so we will not have that.

So, we would say that 1 2 3 4 5 6 7 8 9 10 11 is feasible, which also means that if we have one operator sitting and doing all the assembly of 1 to 1 to 11, then the operator can come with a finished product or an assembly. And if we assume that these tasks times are in minutes, then if one person does it and does it in the order 1 2 3 4 5 6 7 8 9 10 11, then the total time that will be taken is 4 plus 6 10 12 15 16 22 30 32 38 43 plus 4, 47 units of time. Every 47 units we will see that an output comes. Many times what happens is these assemblies, may also require a certain skills like delicate assembly or testing after the assembly is completed and so on.

So, many times if we want to really increase the output of this we need more operators sitting. If 2 operators sit, and if both the operators actually assembled everything. Then we would get 2 pieces in 47 time units, which means effectively one piece in 23 and a half time units, but that would involve for example, creating if activity number 6 is a testing activity. And that would involve creating 2 test benches with the same testing equipment, so if we are going to have a situation where we have several operators sitting, and each one of them doing the entire thing.

Then the output can be 47 divided by the number of operators, but many times such a thing would not be feasible, because that would mean duplicating a lot of resources, that are required for each one of them. Therefore, it is customary in a assembly line balancing, that one operator is given 2 or 3 tasks or certain number of tasks, and the assembly moves to the next operator, who carries out certain number of tasks and so on.

So, that each operator would have a work bench and there will be a time taken by each operator in the work bench. Now, if we assume that we are going to have 11 work benches, where task 1 is done in work bench 1, task 2 in bench 2, task 3 in bench 3, and so on. And if, we put 11 operators one for each and assuming that each operator has the skill to do that activity, then if we start the line with 11 like this.

If, we start the line here at the end of 47 time units one piece will come out, the first piece will come out, but then the second piece would have started by them. And at steady state the output of this with 11 desks and benches or workstations, and 11 operators will

actually be the maximum of the time which is 8 time units. So, if we create a situation with 11 work benches, and one operator doing each, so the output can be one in 8 minutes, as long as we have we do not have parallel workstations.

So, if we have one workstation sequential, and each workstation is carrying out one tasks, so with 11 workstations, if we proceed we will have a an output every 8 minutes and that is call the cycle time, so the cycle time in this case will be 8 minutes. If, we have 11 workstations, but then we have only one operator who first goes to workstation 1 finishes, the assembly takes it goes to workstation 2, finishes the assembly and so on, then the cycle time will be 47 minutes.

So, as long as we do not have parallel stations, the realistic cycle time that we can achieve in this case is anything between 8 minutes and 47 minutes. So, the problem that we wish to solve is: given a cycle time T minutes, what is the minimum number of workstations that we require, what are the tasks that have to be carried out in each of these workstations or what is the allocation of the task to each workstation. Such that the workstation time is less than or equal to capital T, we have minimum number of workstations that is required.

And the precedence relationship that we have described is not violated by the unidirectional flow that we want to have amongst the workstations. Now, that problem is called the line balancing problem, and that problem is also generally solved using heuristic algorithms. So we will see some aspects of line balancing and the heuristic algorithm in the next lecture.