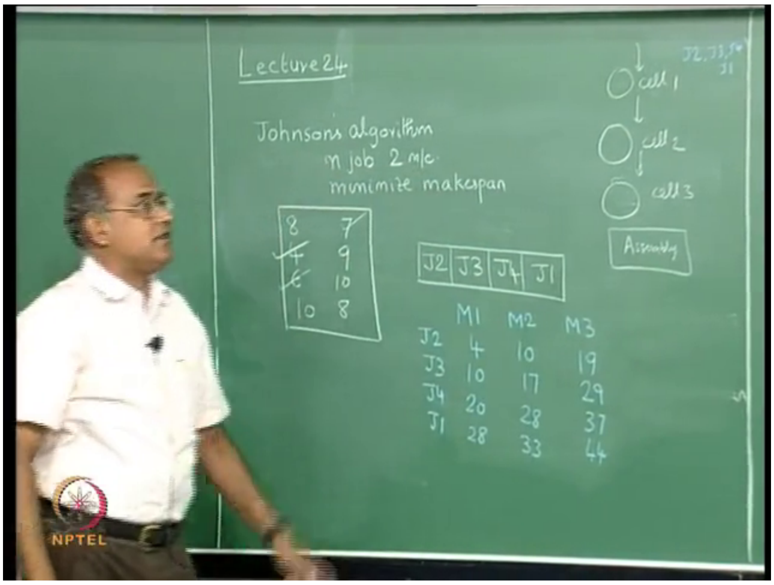
Manufacturing Systems Management
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

Lecture - 24
Cell scheduling and sequencing

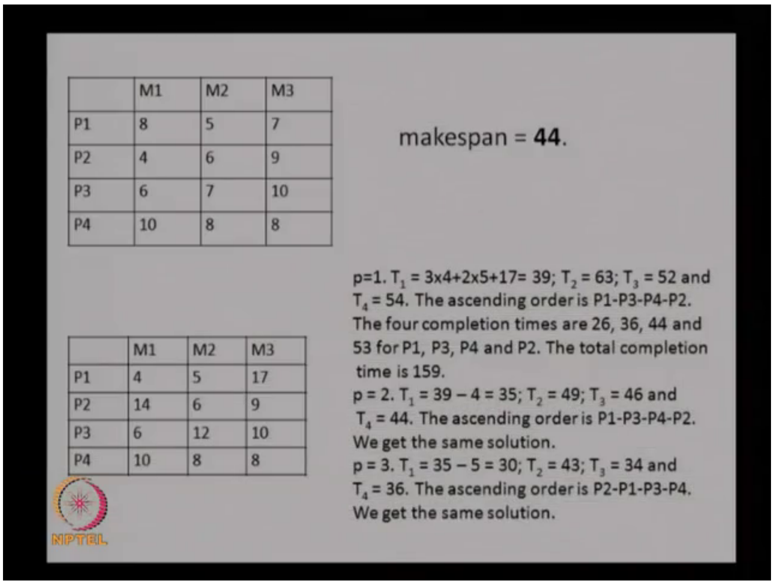
In this lecture we continue the discussion on Sequencing and Scheduling. In the previous lecture we looked at the Johnson's algorithm.

(Refer Slide Time: 00:18)



And Johnson's algorithm which is applied on a n job 2 machine problem to minimize makespan. The problem that we are right now considering is one where there are 4 parts or products made in 3 cells.

(Refer Slide Time: 00:42)

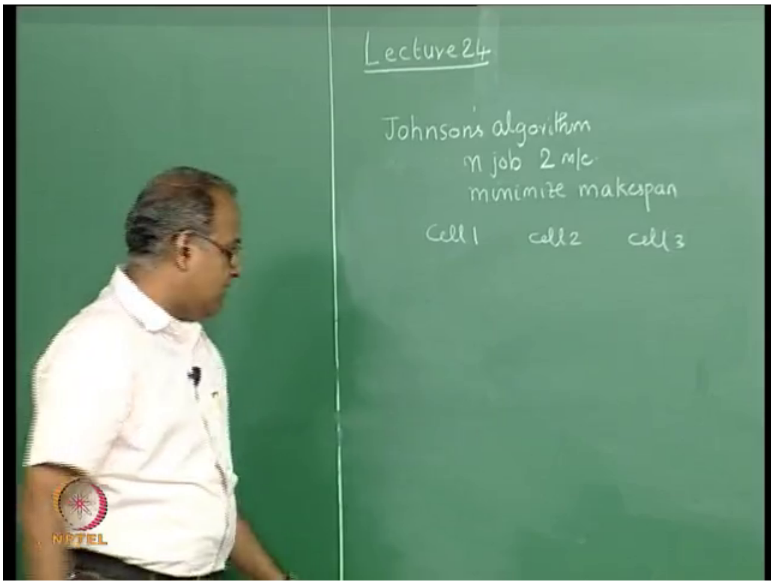


So, the table on the top gives us the processing times while this represents the way in which the manufacturing system is working. So, there are 4 jobs which are J 1, J 2, J 3, J 4; which now have to be sequenced in such a manner they go through all of them go through cell 1 and cell 2 and cell 3 and come here.

So, we are interested in finding out when earliest all of them are available here for the assembly. So, that becomes the makespan minimization problem on 3 machines or 3 cells the Johnsons algorithm that we have seen in the previous lecture looks at solving an n job to machine problem optimally. So, we also defined what is called a Johnson problem a Johnson problem means there are n jobs 2 machines the processing times are given we get the Johnson sequence and then we evaluate the makespan for the Johnson sequence.

Now, how do we apply the Johnsons algorithm; now how do we apply the Johnsons algorithm to the case where we have 3 cells and to the processing time that is given in the other table. So, one of the ways by which we can do that is by solving 2 Johnson problems for example, if we have 3 machines are 3 cells.

(Refer Slide Time: 02:14)



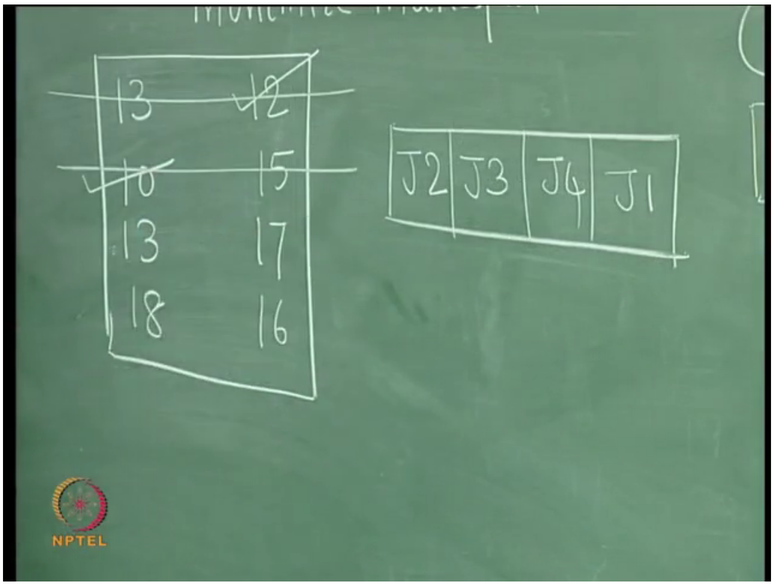
So, we have cell 1, cell 2 and cell 3 we could solve one Johnson problem with 1 and 3 the other with 1 plus 2 and 2 plus 3. So, let us do that and then let us try and find out how the solution looks like. So, we now solve for one and 3 which means I am going to write the; I am going to assume somewhere for the purpose of the solution alone that I am looking at 1 and 3. So, my times will be 8, 7, 4, 9, 6, 10 and 10; 8, now I solve a Johnson problem out of this. So, I will have 4 jobs the smallest one happens here. So, J 2 will come here on the first machine. So, it comes from the left the next smallest is 6; J 3 on the first machine. So, J 3 will come here again from the left.

The next smallest is seven; so J 1 on the second machine. So, J 1 will come from this side and J 4 will automatically come here. So, very quickly the completion times for this particular sequence we have to do this very carefully that we should have now we go back M 1, M 2 and M 3; we now apply this sequence not on this data, but on the other data that is given. So, J 4 comes first J 2 comes first J 2 starts at 0 and finishes at 4 on M 1, it starts at 4 and finishes at 10 on M 2 4 plus 6 10 the third processing time is 9.

So, it starts on M 3 and finishes at 19. So, which means in this sequence the first one will be J 2 and J 2 will be here at 4 here at 10 here at 19 the next one is J 3; J 3 enters here at 4, so plus another 6. So, 10 J 3 second processing time is 7. So, at 10 this is free, so 10 plus 7; 17. So, this finishes at 17, but this was busy till 19 this is like saying J 3 has come here at 17, but this one is still working on J 2 till 19. So, J 3 can get into this only at 19 takes another 10 and finishes at 29.

Next one is J 4; J 4 can enter at 10 J 4 requires a processing time of 10; 10 plus 10; 20. So, J 4 comes out the second one is J 3 third one is J 4; J 4 comes out at 20 here, this is free at 17. So, it enters at 20; 20 plus 8; 28 it comes out, but this is free only at 29. So, 29 it takes another 8 finishes at 37. Now J 1 is the last one J 1 enters at 20 finishes at 28 finishes at 28; now at 28 exactly this machine is free. So, it takes another 5; 33. So, J 1 comes here at 33, but this is free only at 37. So, it takes this at 37 plus another 7; 44 it comes out. So, if we follow this sequence J 2, J 3, J 4, J 1; now at 44 it is completed which is what is shown in the table. So, the algorithm does not stop here the algorithm does something more which is this.

(Refer Slide Time: 06:47)



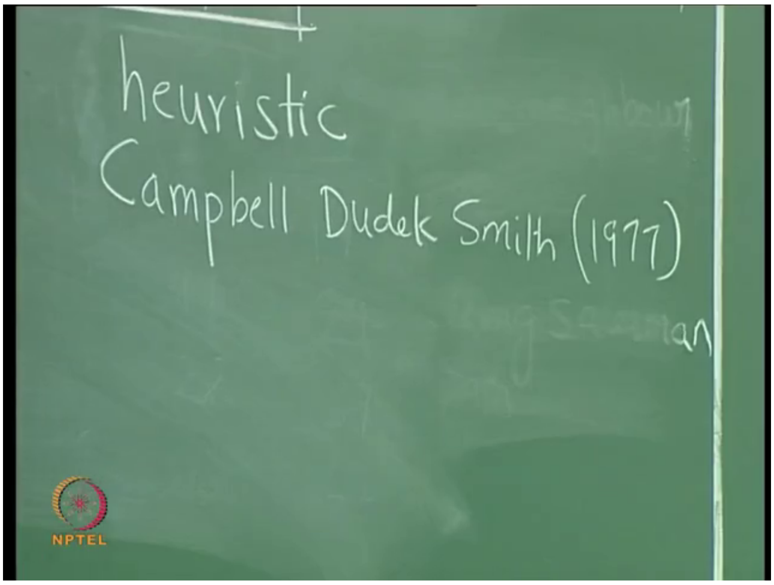
Now, we have done only one part of the algorithm which is to look at M 1 and M 3 which is cell 1 and cell 3; now we have to look at 1 plus 2 and 2 plus 3. So, we look at 1 plus 2 and 2 plus 3 the times are 13 and 12; 1 plus 2 is 8 plus 5; 2 plus 3 is 5 plus 7. So, 13 and 12; 10 and 15; 1 plus 2 is 4 plus 6 which is 10 and 15; 6 plus 7; 13; 7 plus 10; 17 10 plus 8; 18 and 8 plus 8; 16.

Now we solve another Johnson problem out of this the smallest number happens to be here which is for job 2 on the first machine. So, come from the left to right job 2 the next smallest happens here which is for job one on the second machine. So, come from the right and write job one the third smallest happens to be either here or here there are no this is over this is over. So, the third smallest happens to be here, which is J 3 which is again on the first machine. So, J 3 the last one that is available is J 4 which goes here.

So, we get the same sequence and when we apply the same sequence to that data we will get the same 44 it is only incidental that in this particular problem both the junction problems give the same sequence if both the Johnson problems give different if the 2 Johnson problems give 2 different sequences then the makespan computed will be different.

And we will take the better of the 2 solutions in this case both were equal and we say that at time equal to 44 all of them are ready at the assembly if for some reason these 4 are actually 4 parts that go into the same product which is made, then this product can be made at time equal to 44. So, that the makespan helps in the context of a modern manufacturing system using ideas from traditional sequencing and scheduling. Now this algorithm that we have seen is a heuristic algorithm.

(Refer Slide Time: 09:32)

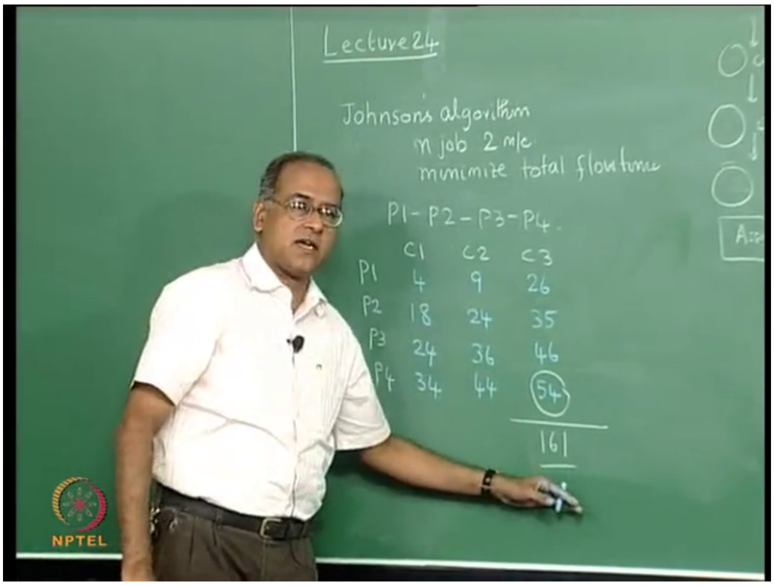


This does not guarantee the best or optimum solution all the time and this is called the Campbell Dudek and Smith heuristic; it is a very popular heuristic to solve makespan minimization problems on flow shops considering more than 2 machines.

If there are 3 machines, we end up solving 2 Johnson problems if there are 5 machines we end up solving 4 Johnson problems and the 4 problems could be M 1 and M 5 M 1 plus M 2, M 4 plus M 5, M 1 plus M 2 plus M 3, M 3 plus M 4 plus M 5; M 1 plus M 2 plus M 3 plus M 4 and M 2 plus M 3 plus M 4 plus M 5. So, like this 4 problems can be created the best out of the 4 is taken as the solution given by the heuristic. So, this is how we apply ideas from flow shop scheduling to solve, but then we will look at one more aspect of flow shop scheduling and then we will see what we do here. So, while the Campbell Dudek and Smith heuristic would help us to minimize the makespan which is under the assumption that all these 4 are actually 4 parts which have to be processed on 3 cells sequentially and come to the assembly. So, 44 or the makespan is the time at which the assembly can start it.

Now, if we want to look at other objectives of minimizing total flow time.

(Refer Slide Time: 11:18)

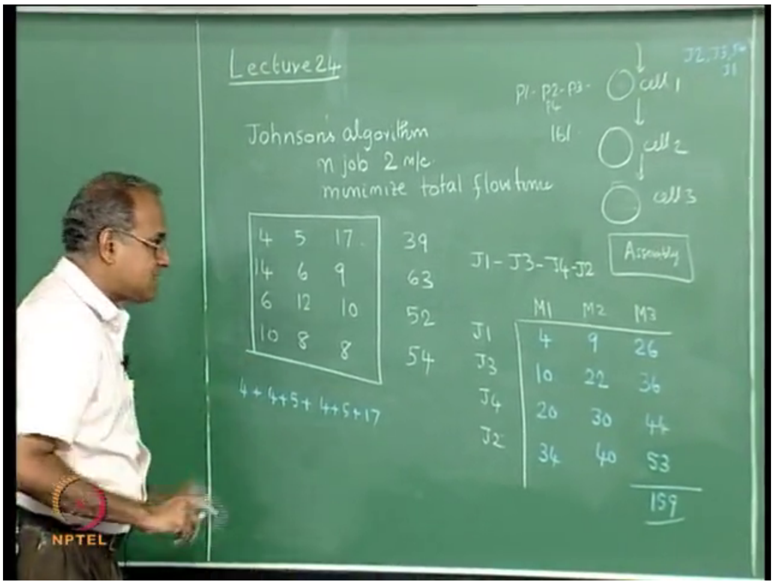


Which means the time taken by all these parts in the system then again we have to solve a different optimization problem because in the makespan we minimize the maximum of the completion times in flow time, we minimize the sum of completion times. For example, if we had followed the sequence P 1, P 2, P 3, P 4 and if we consider the second table given as the processing time data then we look at the computations. So, we have cell 1 or M 1, M 2, M 3 and then if we do P 1, P 2, P 3, P 4; if the sequence is known. Let us quickly do the computation for the second problem. So, P 1 will finish at 4; 4 plus 5;

9; 9 plus 17; 26; P 2 will enter at 4 finish at 18. Now 18 plus 6; 24, but can take it again at 26 plus 9; 35 P 3 will start at 18 finish at 24; it is free at 24; 24 plus 12; 36. So, this is free at 35, but it can still do at 36; 36 plus 10; 46.

Fourth one 24 plus 10; 34 machine is free at 36; 36 plus 8; 44; 46 plus 8; 54. Now in this case makespan is 54 sum of completion times which is the extent of inventory that is being held will be 26 plus 5; 26 plus 35 plus 46 plus 54 which is 161; 161 is the sum of completion times or flow time in the system. So, if we use the sequence.

(Refer Slide Time: 13:54)



J 2, J 1 or P 2, P 1, P 2, P 3, P 4 sum of completion times is 161; now let us look at another heuristic solution to solve this problem. Now what we do in this case is we try and first compute a certain weight associated with each part. So, if we look at part P 1 if you look at this data 4, 5, 17, 14, 6, 9, 6, 12, 10, 10, 8, 8.

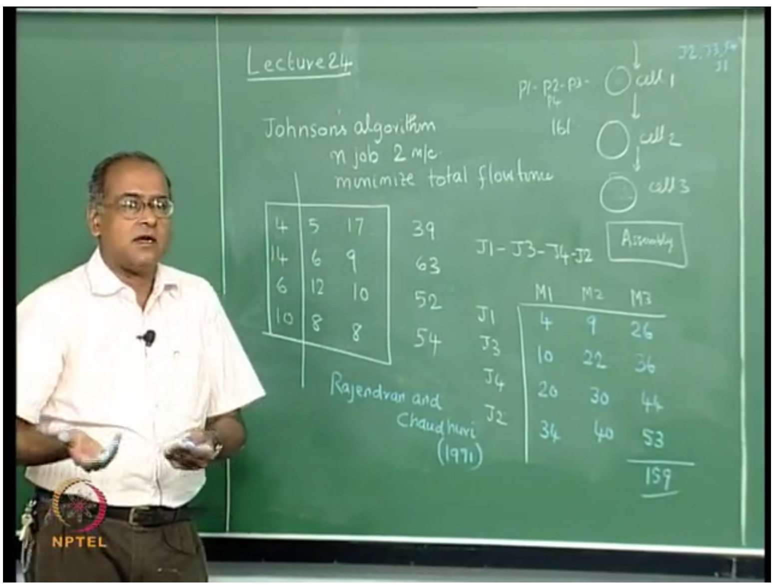
Now, what we do here is there are 3 machines. So, we calculate an index which is like 3 into 4 plus 2 into 5 plus 1 into 17. So, this is 3 into 4; 12 plus 10; 22 plus 17 is 39; we compute another index 3 into 14; 42 plus 2 into 6; 12; 54 plus 9; 63 another index 3 into 6; 18 plus 2 into 12; 24 plus 18; 42 plus 10; 52 and the index 3 into 10; 30 plus 2 into 8; 16; 46 plus 8; 54. So, the index or the 4 indices are 39, 63, 52 and 64 and we arrange them in increasing order to get the sequence J 1, J 3, J 4 and J 2 or P 1, P 3, P 4 and P 2; now the completion times for this sequence J 1, J 3, J 4, J 2 on M 1, M 2, M 3; this is the notation from scheduling normal flow shop scheduling in our context this will become P 1, P 3, P 4 and P 2 this will become C 1, C 2 and C 3.

Now, the completion times first is this. So, finishes at 4; 4 plus 5; 9; 9 plus 17; 26. Now comes J 3 finishes at 4. So, can enter at 4; 4 plus 6; 10 by 10; M 2 is free. So, 10 plus 12; 22 has to wait till 26; 26 plus 10; 36 J 4 starts at 10 takes another 10 finishes at 20 waits till 22 takes another 8; 30 waits till 36 takes another 8; 44; the last one is J 2 starts at 20 finishes at 34, again this is free at 30 itself. So, 34 plus 6; 40 wait still 44 and finishes at 53.

Now sum of the completion times will be 12; 16 plus 3, 19; 6, 10 159 which is better than the 161 that we got now what is the rationale behind this and why are we computing an index here and then multiplying it and then getting something the reason, we end up doing that is if we actually find this computational time which is 26. Now this is 4; 4 plus 5; 9. So, this 26 can be written as 4 plus 4 plus 5; there is a 9 that comes out of 4 plus 5 the 17; this 26 comes out of 4 plus 5 plus 17. So, the last number is essentially 4 plus 5 plus 17, but the fact is it has been arrived at this kind of a sequence.

So, somewhere when we do this expansion if you add all these terms which is in some sense time spent heuristically, then 4 comes 3 times 5 comes 2 times 17 comes 1 time the second part of this algorithm; this is one such solution the next solution will be to look only at.

(Refer Slide Time: 19:05)

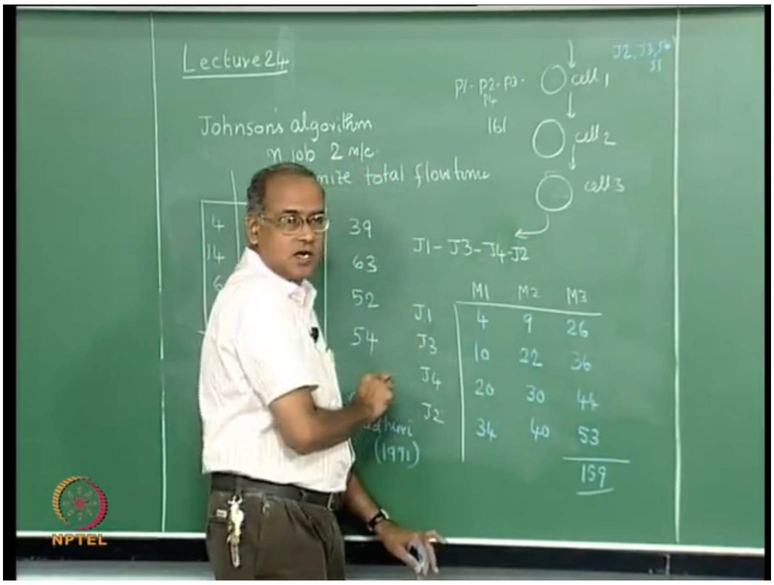


This to look only at this now this will become this is 39 minus 4 will give us 35; T 2 is 63 minus 14 which is 49 for the third one 52 minus 12 minus 6; 46 and 54 minus 10; 44 we get the same sequence and the same solution and the last one would be to look at 39 minus 9 which is 30; 63 minus 20 which is 43; 52 minus 18 which is 34 and 54 minus 18 which is 36; once again we get the same solution. So, essentially the idea is that these when these jobs visit these machines the time spent gets compounded. So, time spent on this would be 4 on this is 9 the time at which it comes out time at which it comes out is 26 one way of saying is 26 is actually 4 plus 5 plus 17; there is no need to multiply this by 3 this by 2 and this by 1, but if we look at it very very carefully we see somewhere that this 4 will be contributing more than what this 5 would be contributing to the completion.

Therefore, we define an index where by 4 is multiplied by a bigger number 5 is multiplied by a number that just smaller and 17 is multiplied by 1 to get these indices and then we sort them based on the indices. So, this is another heuristic method to look at the flow time problem and this heuristic is from Rajendran and Chaudhary; there are other heuristic methods that are also available; we just introduced one heuristic method to solve each of these cases the reader or the student can look at several other text books or other related literature to find out that there are other heuristics available each of these heuristics perform well under certain conditions on under a certain test problems. There could be test problems where this heuristics performs better than other heuristics. And there could be other problem instances or test instances where some other heuristic proposed by another author could fair better than this heuristic the student has to keep that in mind while the optimum solution will be the same irrespective of the method that we use.

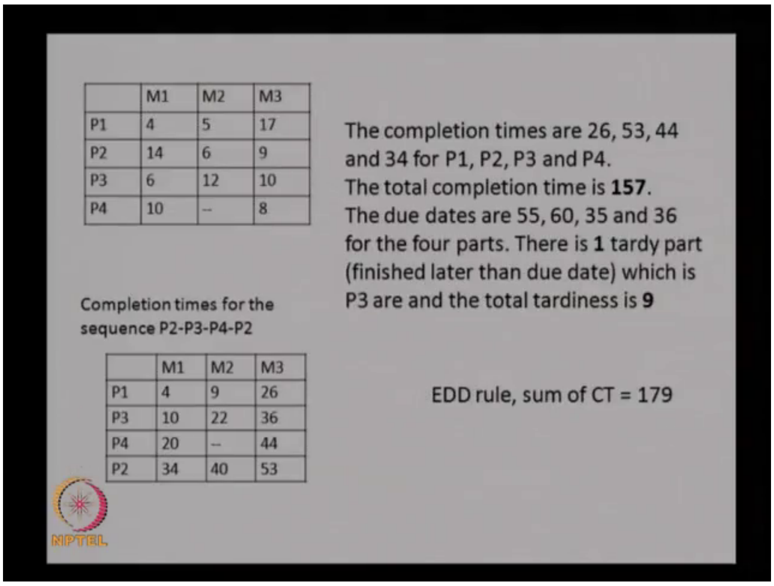
The same problem can be solved as a binary integer programming problem or using a different branch and bound algorithm to get the best value of this, but when we use heuristics each algorithm can give a different solution sometimes more than one algorithm gives the same solution. So, this is how we look at these objectives; now the flow time objective essentially says that if each of these 4 jobs do not go to the assembly or they complete processing on 3 cells.

(Refer Slide Time: 22:32)



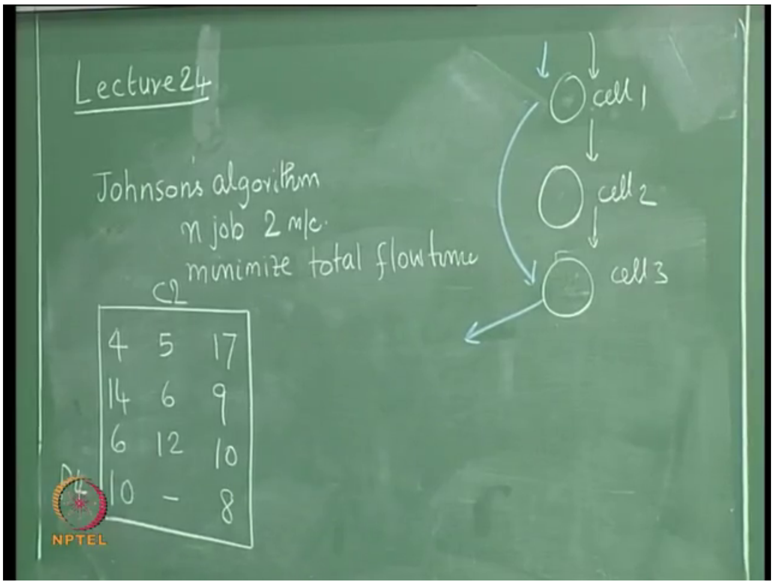
And leave the shop to an assembly somewhere else then what is that time that they have spent in the system if they follow the sequence 1, 3, 4 and 2, then the total time that they have spent in the system is 159 time units. Now let us look at another instance.

(Refer Slide Time: 23:05)



Where we have data which is shown on the first part of the table let me reproduce this data again here.

(Refer Slide Time: 23:18)



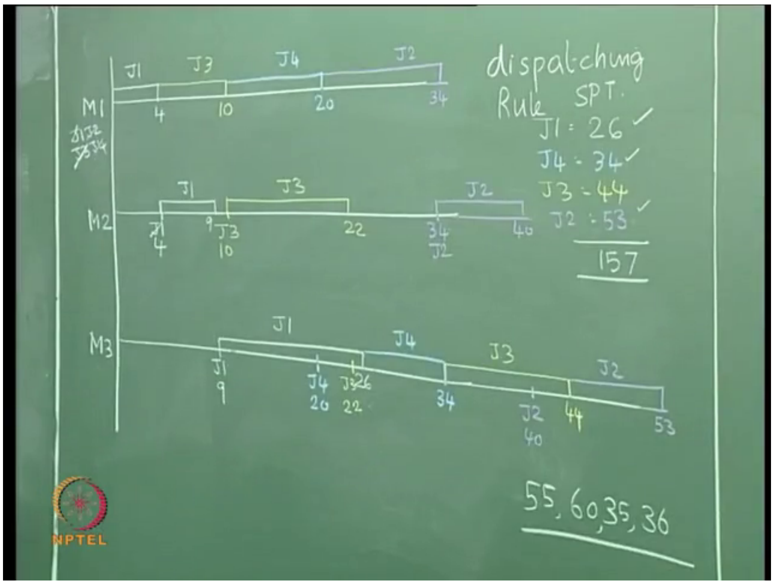
So again, there are 4 jobs which are on 3 machines the times now are 4, 5, 17, 14, 6, 9, 6, 12, 10 and 10 dash 8.

Now, the only difference between this type of data and the previous data is that there is a dash here which meant that part 4 does not visit cell 2 while all other parts visit the cells in the same order. Now here part 4 alone goes to cell 1 and then goes to cell 3 and then comes out it does not visit cell 2. So, it violates the flow shop assumption that all the jobs will visit all the machines. So, we do not apply flow shop algorithms at this for this situation. So, we could apply typically what are called the job shop algorithms to do that.

So, we now assume that each job has a unique order of visit of the machines job one will visit the machines are the cells in the order cell 1, cell 2 and cell 3, cell 1, cell 2 and these 3 are the same this is cell 1 and then directly to cell 3. So, we now treat this as a job shop scheduling problem even though there are some other ways by which the literature has handled it or authors have handled these kinds of problem situations; we would now use this as a way to also look at how we model it as a job shop when each of these parts. Now has a unique sequence 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 3 as a unique sequence.

Now, we apply the idea of using Gantt charts.

(Refer Slide Time: 25:34)



So, this we would call as cell 1 or machine 1; machine 2 and machine 3, we would use this. So, now, all the 4 jobs visit machine 1. So, all the 4 jobs visit machine 1. So, all 4 jobs are here. So, J 1, J 2, they are all here on machine 1. So, we take the one that has the let us use the shortest processing time rule and create this chart. So, the shortest processing time is for P 1 or J 1; so on M 1. So, let us use this. So, J 1 goes in at time equal to 4 J 2 J 1 finishes on M 1 and goes to M 2

So, this is J 1 enters at time equal to four. So, at time equal to 4 on M 1 we have J 2, J 3 and J 4; so J 2, J 3 and J 4. So, J 3 goes next. So, J 3 goes next. So, let us use this for J 3. So, J 3 finishes at 10 and J 3 will come here at 10. Now again at time equal to 4 J 1 is here on M 2. So, J 1 takes another 5 units J 1 finishes at 9 and J 1 moves to M 3 at 9. Now at time equal to 10 J 2 and J 4 are available 14 and 10. So, J 4 comes based on shortest processing time. So, let us use this color. So, from 10 to 20; J 4 finishes at time equal to 20 and J 4 will now come directly to M 3 at time equal to 20.

Now, at time equal to 9 J 1 is here on M 3; so at 9 J 1 starts on M 3. So, 9 plus 17; 26 J 1 finishes at 26. So, J 1 finishes at 26 and comes out now at time equal to 10; J 3 is available here on M 2 and J 3 on M 2 is 12. So, this starts at 10 and goes on till 22. So, J 3 is available on 22. So, J 3 will come here at time equal to 22. So, we again move now and then we realize that we are here at time equal to 20; this is busy till 22, this is busy till 26.

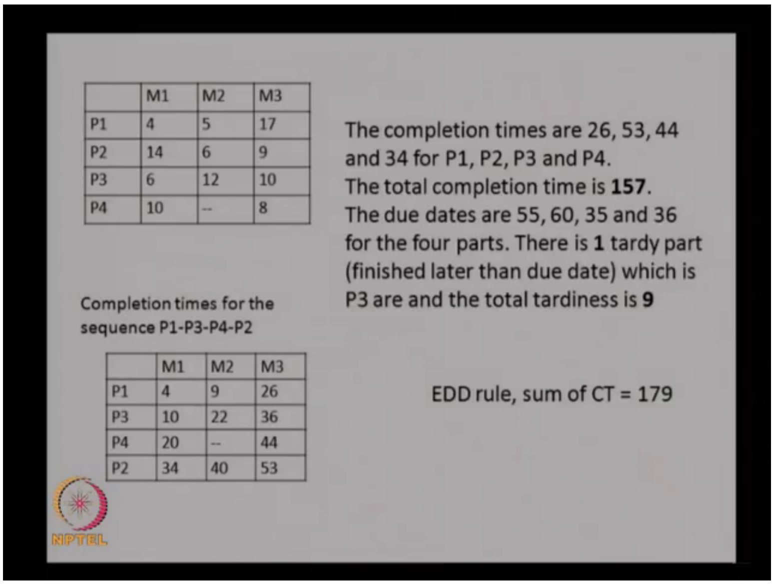
So, we are at time equal to 20; J 2 is the only job that is remaining. So, J 2 takes another 14. So, J 2 finishes here at 34 and 34; J 2 comes here. So, we now go back to this chart here we are at time equal to 22, but then J 2 is going to come only at 34. So, at now all these 4 jobs are over here. So, J 2 is going to come only at 34. So, nothing we cannot do anything; now we are here at time equal to 26 while J 4 has come at 20; J 3 has come at 22; so on M 3, J 4 and J 3.

We are now choosing J 4 based on smaller processing time not because J 4 came earlier the rule that we use consistently is if there is a machine that is free and if there are jobs waiting in front of it our rule is to choose the job that has the smallest processing time irrespective of when it arrived, it is incidental that J 4 which arrived earlier actually has the smaller of the processing times. So, J 4 comes now. So, J 4 starts at 26 finishes at 34. So, J 4 finishes at 34. So, J 4 finishes at 34.

Now, again we are here at time equal to 34 J 2 is available at time equal to 34 here. So, J 2 on M 2 is another 6 which is 40. So, 40 and at 40; J 2 comes in at 40; now we are at time equal to 34 J 3 is the only job available here. So, J 3 takes another 10 minutes. So, J 3 finishes at 44 and J 3 finishes at 44; now J 2 has come at 40; 0 J 2 requires another 9 units; so 44 plus 9 53 J 2. So, J 2 finishes at 53.

Now, we can see the difference here that when we applied this kind of a job shop scheduling idea we get a schedule where the based on shortest processing time J 1 finishes at 26, but there we have computed these processing time for the sequence P 1, P 3, P 4 and P 2 we have computed it for the sequence P 1, P 3, P 4 and P 2 we get.

(Refer Slide Time: 32:59)



We get for P 1 we get 26 for P 3 here we have 44; there it is 36 for P 4 there it is 44, here it is 34 and it is 53. So, other sum of completion times will be 10; 14 plus 3; 17; 3, 6, 4, 157 as against 1; 159; we could get from the earlier sequence. So, there are some modifications that have to be made if we want to use the flow shop idea to try and get this, but then the moment we have a situation where a particular job skips a machine it is good to treat it as a job shop scheduling problem apply the Gantt charts this is the Gantt chart for the job shop scheduling problem where the schedules are shown for each of these jobs we have used 4 distinct colors for each of these jobs. And we have written the solution for the Gantt chart.

Now Gantt chart is more of a evaluative kind of tool were given a rule the chart gets defined. So, there is not much of an optimization here the rules that we have used consistently are if there is only one machine one job if the machine is free if there is only one job waiting in front of it we will load that job we will not keep the machine idle that is called a non delay schedule we will not keep a machine idle in the hope that some other job that is going to come later if that is done first we can get a better answer.

So, when a machine is free and if there is one or more jobs waiting the machine will not be kept idol if there is more than one job waiting the job is chosen based on what is called a dispatching rule and the dispatching rule that we have chosen is called the SPT rule or the shortest processing time rule. So, we have used the dispatching rule SPT to get this Gantt chart, we could use other dispatching rules to get different Gantt charts for example, if the 4 jobs have due dates which are 55, 635 and 36; if the due dates are 55, 635 and 36.

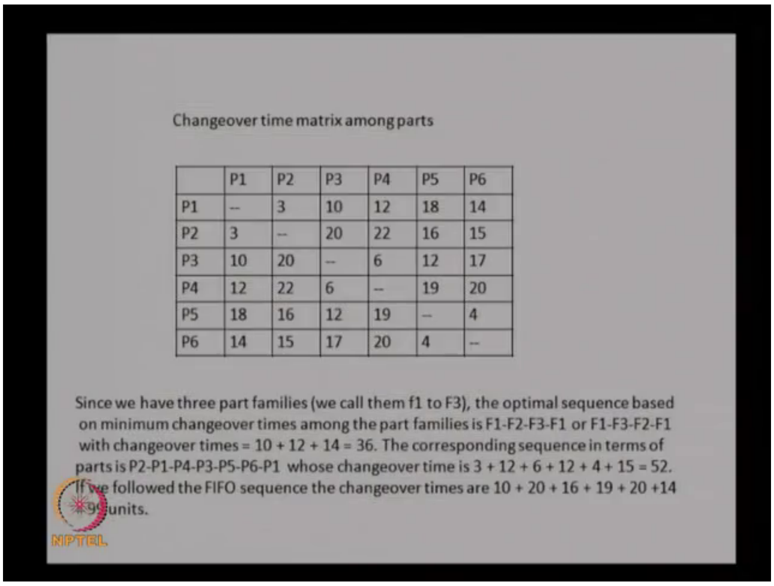
And instead of using the shortest processing time rule we used what is called the earliest due date rule which means right in the beginning all the 4 jobs were available on M 1 the times were 4, 14, 6 and 10. We chose job one because it had the smallest processing time if we were to choose one of these 4 based on the earliest due date the earliest due date is here for J 3 and we would have chosen J 3 first instead of J 1 and the entire Gantt chart would have been very different.

So, when a particular job skips a machine or skips a cell it is advisable to try and use the job shop idea to try and get a Gantt chart and get a solution based on the dispatching rule, but dispatching rules are also a heuristic in nature and dispatching rules do not and need not guarantee optimality; optimality of the job shop scheduling problem is also difficult to reach they are usually not solved as 0, 1 integer programming problems branch and bound algorithms have limited scope if you want to find the optimum solution to the job shop scheduling problem they have limited scope most of the times they are solved using dispatching rules which are easy to understand and easy to implement and consistent though there are other heuristics that are available to solve job shop scheduling problems.

The Gantt chart and the dispatching rule is a very convenient way to evaluate a solution now the Gantt chart is here if we want to evaluate the makespan the makespan is the maximum of these which is 53, if we want to compute the total flow time some of these 4 times or sum of completion times is 157; if the due dates are given this way job ones due date is 55, it completes at 26, it is early job 2 due date is 60, it completes at 53, it is early job 3 due date is 35, it finishes at 44. So, it is tardy with tardiness equal to 9 job 4 due date is 36 it finishes at 34 it is early.

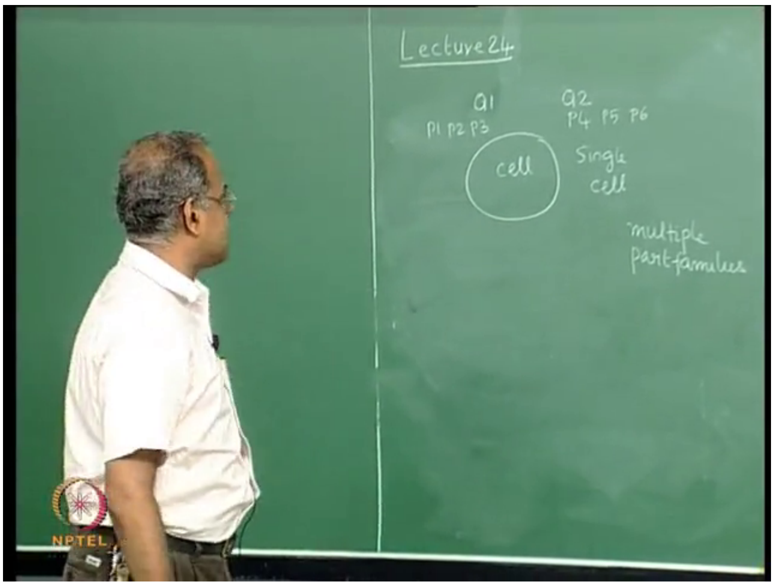
So, 3 out of the 4 jobs are early one job is tardy the tardiness equal to 9 earliness can be calculated as based on 26 and 55; there is a 29 earliness job 4, there is an earliness of 2 thirty one job 2 there is an earliness of 7. So, 38 is the total earliness that we have. So, given the Gantt chart and given the dispatching rule the Gantt chart can be drawn and all other objectives can be evaluated using this Gantt chart. So, this is how we use ideas from flow shop and job shop scheduling in the context of cellular manufacturing.

(Refer Slide Time: 39:10)



We look at another aspect of sequencing and scheduling in the context of cellular manufacturing. Now this is a very interesting idea that let us assume that we have a single cell.

(Refer Slide Time: 39:29)

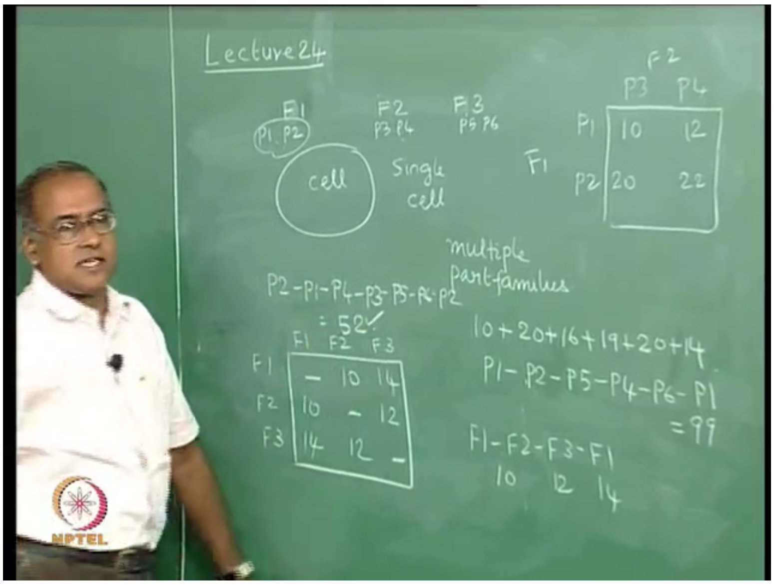


Single cell and this single cell processes multiple part families in the cell formation that we spoke about we created cells such that each cell has an associated part family.

Now, we look at a situation where we could have multiple part families for example, the same cell could be used to make products P 1 and P 2 or let me call as Q 1 and Q 2 and we could say that there are some variants of this and these variants could be P 1, P 2, P 3, here and they could be P 4, P 5, P 6, here. So, in such cases how do we optimize or find out the order in which these parts have to be given, now if we have a situation like this there is a changeover matrix that is given here there are 6 parts P 1 to P 6, but then we have made an assumption that P 1 to P 3 is one part family and P 4 to P 5 is another.

Now if we look at this we actually have a 6 by 6 matrix with P 1, P 2, P 3, P 4, P 5, P 6 as 6 parts and the way this is written is let us say there are 3 part families.

(Refer Slide Time: 41:39)



Which we call Q 1, Q 2, Q 3 and this contains P 1, P 2; this contains P 3, P 4; this contains P 5, P 6.

Now, in the matrix notation we have used F 1, F 2 and F 3 as 3 families which this cell is processing; now suppose on a particular day when we begin processing on this cell, if we have a sequence which is if we have a different sequence with say a given sequence that we have. So, if we had a fee 4 sequence which is first in first out sequence then we could have 10 plus 20 plus 16. So, let me write this. So, 10 plus 20 plus 16 plus 19 plus 20 plus 14 which means we have a sequence 10 plus 16 which is P 1, P 2, P 1, P 5, P 1, P 3, 10 plus 20 plus 16. So, P 1 plus P 2 plus from then we move to P 5 to P 4 and from P 4, we move to P 6 and P 6 to P 1. Let us say the parts have arrived in this order P 1, P 2, P 5, P

4, P 6, P 1, they have arrived in that order. So, if we use a first in first out heuristic, then the total of changeover times will be this and that is given by 99 units.

But then we look at part families here there are 3 part families. So, P 1; P 2 is one part family P 3, P 4 is another part family P 5, P 6 is the third part family. So, we should look at a part family based scheduling which could give us P 2, P 1 and from there P 4, P 3 from there P 5, P 6 and then we go to P 2. So, this would give us a changeover time of 52. So, what we are essentially trying to suggest here is that there are 3 part families each part family has certain number of parts now within these we have shown only 2 parts. So, whether you do in the order P 1, P 2 or P 2, P 1, they will be, but if you have multiple parts if we have 4 or 5 parts within the part family or variations within a product, then we could solve a smaller travelling salesman problem inside to find out the best sequence, we have seen that in the previous lecture as to how we solve the travelling salesman problem, then what we can do is we could now solve a travelling salesman problem amongst the apart families themselves we could solve a TSP among the part families themselves. So, we could go back to this and create a 3 by 3 tsp with F 1, F 2, F 3, F 1, F 2, F 3.

So, if we look at F 1 this comprises of P 1, P 2, F 2 comprises of P 3, P 4. So, if you look at P 1, P 2 data and P 3, P 4 data the minimum changeover that we can think of is between 1 and 3 which could be 10. Now we look at F 1 and F 3 now we look at this data between P 1, P 2 and P 5, P 6. We realize that there is a 14. So, there is a 10 there is a 14 now F 2 F 3 would give us between P 3, P 4 and P 5, P 6 there is a 12. So, first we solve a travelling salesman problem to find out the sequence in which we will schedule the families. So, in this case the schedule will be F 1, F 2, F 3, F 1. So, this is 10 plus 12; F 1, F 3, F 2, F 1 would also give F 1, F 2 as 10; F 2, F 3 is 12; F 3, F 1 is 14 because we have only 3 families we will get only one solution.

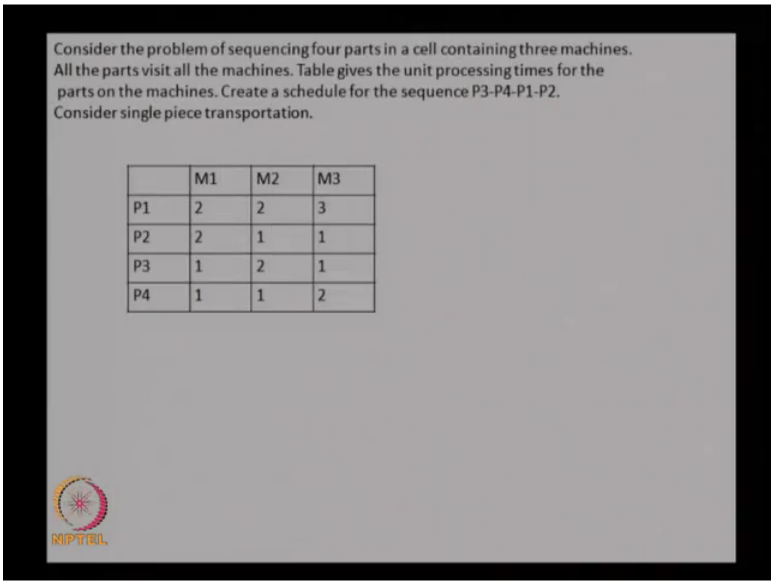
If we had large number of families we could solve a travelling salesman problem amongst the families to find out the order in which we take the families we have already seen in a previous lecture as to how to do it the only difference is here we take the minimum changeover time to get this. For example, if we go back to that matrix, if we look at F 1 and F 2; F 1 is P 1, P 2, F 2 is P 3, P 4. So, we look at the small matrix which is between P 1, P 2 and P 3, P 4 which is the matrix between F 1 and F 2; if we go back we realize that the values are 10, 12, 20 and 22.

Now as a representative measure we take the minimum of these values sometimes, we could even take the average of these values or we could even take the maximum of these values. So, depending on what we consistently use we will get a part family sequence and then within every part family we have the parts within a part family which are sequenced based on a travelling salesman. So, this would ensure that the items are taken in the order such that within a part family we follow that rule amongst the part families; we follow a certain rule and since all of them are based on minimum changeover. So, at the end a sequence where we try and do this kind of a thing where this is F 1, F 2, F 3 and back to F 1. And then within that the sequence would give us a small smaller changeover time compared to a first in first out their part families from parts belonging to different families can arrive and a first in first out can give us a not so good solution.

The strength of cellular manufacturing lies in the fact that part families are created with parts which are similar. Therefore, scheduling within a part family even though we solved it as a travelling salesman problem scheduling within scheduling parts within a part family would still give us a sequence with minimum changeover now scheduling the part families themselves, because part families themselves could be different and the changeover times amongst the part families can be large.

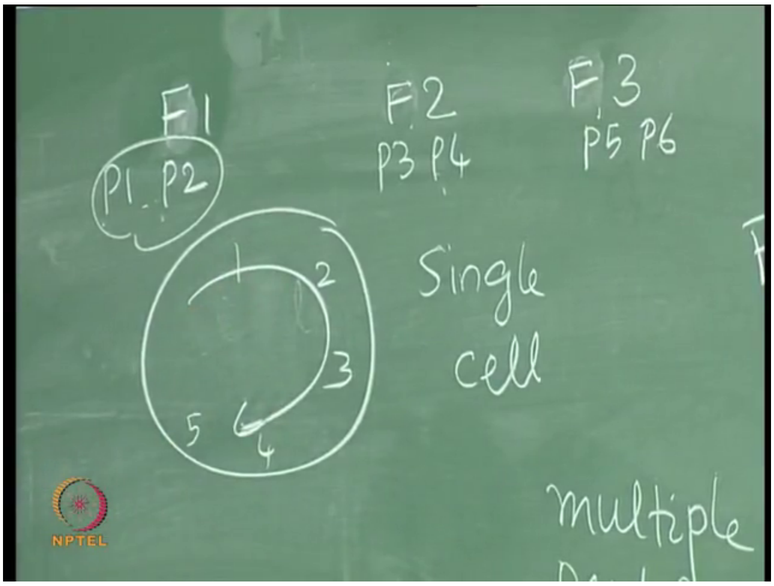
So, part families have to be scheduled. And within the part family we will have the items that are being scheduled. So, the basic idea to exploit the part family concept is to try and schedule parts within a part family together. So, that the changeover times can be minimized that is best brought out through this particular example.

(Refer Slide Time: 50:20)



The last part that we will see in the cellular manufacturing sequencing and scheduling is to exploit the advantage of small transportation batches. So, when we have a manufacturing cell which let us say has.

(Refer Slide Time: 50:45)



If we have a manufacturing cell that has machines like this 1, 2, 3, 4 and 5, these machines are located close to each other and therefore, it is possible to have what is called a single piece transportation within the manufacturing cell. In a traditional manufacturing system with department specializations, these machines could be located in different places that would necessitate a large batch transfer. Cellular manufacturing has transportation in single piece. So, single piece transportation can bring down the time taken to produce.

So, this aspect we will look at through an example in the next lecture.