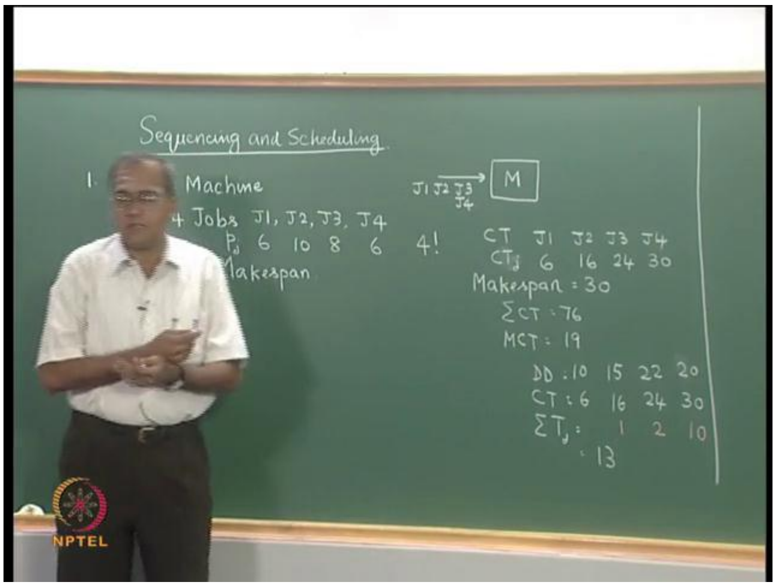
Operations and Supply Chain Management
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

Lecture - 25
Single Machine Sequencing. Two Machine Flow Shop – Johnson’s Algorithm

In the lecture, we continue our discussion on Sequencing and Scheduling. In the previous lecture, we had seen the assumptions, the objectives.

(Refer Slide Time: 00:18)



and different types of sequencing and scheduling problems. We will now address each one of the problems in detail, and see how we can optimize or minimize the chosen objective. So, the first problem that we will be looking at is a single machine sequencing problem. We take a numerical example, to explain few ideas in single machine sequencing. So, we consider situation with 4 jobs, which we call J 1, J 2, J 3, and J 4. So let the processing times be 6 10 8 and 6 units of time, these units of time can be minutes, can be hours and so on, so it is customary to keep them as minutes.

Now, we want to try and sequence them send them on a single machine, where the single machine is here and we want to send them, for processing on the single machine, so that we try and optimize or minimize a chosen objective. Now, there are 4 jobs J 1 J 2 J 3 and J 4 and these 4 jobs can be arranged in 4 factorial ways, in general, if we have n jobs, they can be arranged in n factorial ways. Now, depending on the objective, one or more of these n factorial sequences will turn out to be optimum. Now let us look at minimizing makespan.

Now, first let us explain the computation of completion times and then we try and look at each of the objectives, now if this is a machine and we try and send them in the given order say J 1 J 2 J 3 and J 4, now J 1 enters at time equal to 0. So, the completion time for J 1 J 2 J 3 and J 4, J 1 will enter at time equal to 0 and come out, at time equal to 6, so J 1 comes out at time equal to 6. One of the assumptions is that the machine can process only one job at a time. So, the machine becomes available at time equal to 6 and then J 2 goes and comes out at 6 plus 10 equal to 16 time units J 2 comes out.

Now at time equal to 16 the machine is free. So, J 3 enters, so 16 plus 8, 24 is the time J 3 comes out, so at time equal to 24 the machine is again free J 4 enters takes another 6 units of time comes out at time equal to 30. So, if we consider the sequence 1 2 3 4, which is the order, in which the jobs are going inside, now the jobs come out at time 6, 16, 24 and 30. Now, as I said there are two important assumptions one is the machine is not idle at all and a machine can process only one job at a time.

So, we also make an assumption that the machine is not idle or the machine is not kept idle, when there are jobs waiting in front of it. Now if we look at Makespan as the objective Makespan is the time, at which the last of the jobs is completed. So, Makespan for this sequence is 30, for a given sequence J 1 J 2 J 3 J 4, the sum of completion time is 6 plus 16 plus 24 plus 30, which is 22 plus 426 46 plus 30 76. So, sum of completion times equal to 76 and mean completion time equal to 76 divided by 4, which is 19 is the mean completion time.

So, given a sequence J 1 J 2 J 3 J 4, we can now calculate the Makespan, we can calculate the mean completion time, let us assume that, the due dates for these 4 jobs are 10 15 22 and 30. So, if we follow the sequence J 1 J 2 J 3 and J 4, the completion times are 6 16 24 and 30, we will assume that the due dates are 10 15 22 and 20 for these four jobs. So, the completion times are 6 16 24 and 30, now job one comes out at time equal to 6, it is due date or time, at which its due is 10, therefore it is early, job 2 comes out at equal to 16, it is the due date is 15.

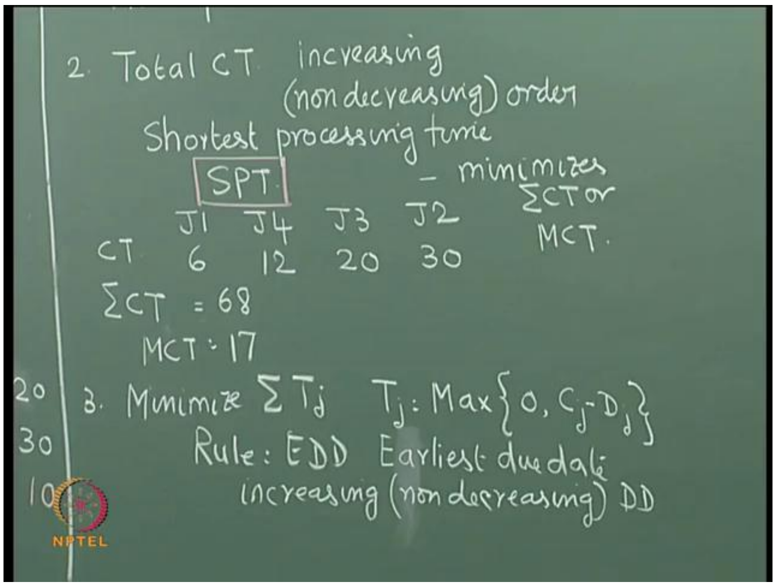
So, the tardiness is one here the completion time is more than the due date, so tardiness is one here the tardiness is 2 and here the tardiness is 10, so total tardiness is equal to 13, for this So, what we have right now done is for a given sequence, we have explained, how we calculate the 3 objectives, once the completion times are calculated all the objectives now depend on the completion times, some of them depend on the due date.

So, the given process times, you would call this as p j process time of job, we have now given the processing time p j of job j, we have now computed here the completion time of job j, once the completion time and due dates are known, we can calculate the objectives. Now, your optimization problem is what is the sequence that minimizes Makespan? What is the sequence that minimizes, mean completion time or sum of completion times? What is the sequence that minimizes total tardiness and so on, we will now address these objectives in a little more detail.

Now, as long as we do not keep the machine idle and as long as the machine can process only one job at a time, whatever order we sent these jobs, the last of the completion times is going to be the sum of the processing times, which is 30. Whether we send it in the order 1 2 3 4 or whether we send it in the order 2 1 3 4 or any one of the 4 factorial ways by which, we can send this job, the Makespan is going to remain the same.

In fact, the assumption that the machine will not be kept idle, when there is a job waiting in front of it, would actually help the Makespan or alternately, if we are looking at Makespan as the objective then does not makes sense for us to keep the machine idle, when there are jobs waiting in front of it. So, as soon as the jobs, all the jobs are available at time equal to 0, so the machine starts processing 1 job after another and in whatever order, we send these jobs, the total Makespan is going to be 30, so Makespan is equal to 30.

(Refer Slide Time: 09:00)



2. Total CT increasing (non decreasing) order

Shortest processing time

$\boxed{\text{SPT}}$ – minimizes ΣCT or MCT.

	J1	J4	J3	J2
CT	6	12	20	30

$\Sigma CT = 68$

MCT = 17

20
30
10

3. Minimize ΣT_j $T_j : \text{Max}\{0, C_j - D_j\}$

Rule : EDD Earliest due date

increasing (non decreasing) DD

So, minimizing Makespan on a single machine is not a very difficult optimization problem, any order any one of the n factorial sequences will give us the same Makespan, which is equal to sum of the processing times. Then we move to the total completion time or mean completion time, now we ask ourselves the question, if we send the jobs in this order 1 2 3 4, the sum of completion time is 76 and the mean completion time is 19, what is the order, in which we have to send these jobs, such that sum of completion times is minimized.

Mean completion time is sum of completion times divided by the number of jobs, so whether we minimize sigma C T or whether we minimize M C T, it means the same, because you are dividing by n and in this example, we are dividing by 4. So, what we want do is if we want to minimize this 76, now let us go and find out, how we got this 76, 76 was got by the sum of 6 plus 16 plus 24 plus 30.

Now how did we get this 6 16 24 and 30, now the moment, we chose an order 1 2 3 4, 6 is the processing time of the first job 16 became processing sum of the processing times of these two, 24 becomes sum of the processing time of these three and 30 becomes sum of the processing times of these four. So, essentially we want to for the order sequence, such that sum of these numbers are minimized, now when we want to minimize the sum of these 4 numbers, it only makes sense to minimize all the individual numbers.

Because the first job that we pick, which is, which has processing time 6 finds, it is place in here, here, here and here, so whatever job, we fix as the first job, it is processing time is going to contribute to all the 4 completion times, whatever we pick as the second job in this example 10. Now, it is contribution is going to be from here and here, whatever we pick as the 3rd job, it is contribution will be from here and here, whatever is the 4 job will contribute here.

So, if we want to minimize the sum of completion times, we wish to minimize all these 4 of them and whatever, we are going to chose here is going to contribute 4 times, whatever we are going to fix here is going to contribute 3 times and so on. Therefore it is only correct that, we use the smallest of them first, which contributes in all the 4 completion times, then next smallest comes second, which contributes in 3 of the completion times and so on.

Therefore, we arrange the jobs in increasing order of processing times and in general non decreasing order of processing time and this is called shortest processing time also called as S P T. So, arrange the jobs in the increasing or non decreasing order of processing times to get the sequence, now the processing times are 6 10 8 and 6. So, we arrange them in the order J 1 J 4 J 3 and J 2, now the moment, we order them in J 1 J 4 J 3 and J 2, the completion times are C 1 has a processing time of 6.

So, this is complete at time equal to 6, at time equal to 6, the machine takes up J 4 takes another 6 units of time do finishes at 12, at 12 it takes up J 3 takes another 8 units of time finishes at 20 and at 20, it takes up the last job, which is J 2, which has a processing time of 10 and this will finish at time equal to 30. Now, sum of completion times is 6 plus 12 18, 18 plus 20 is 38, 38 plus 30, 68 and mean completion time is 17, in this example, the optimum value of mean completion time is 17.

So, the shortest processing time rule minimizes sigma C T or M C T mean completion time, this is an extremely important result, which finds it is way into other sequencing and scheduling problems. More importantly this is introduced us to the rule, which is called shortest processing time rule, where we try and arrange the job to form a sequence, such that the processing times are in increasing order or non decreasing order as in this example.

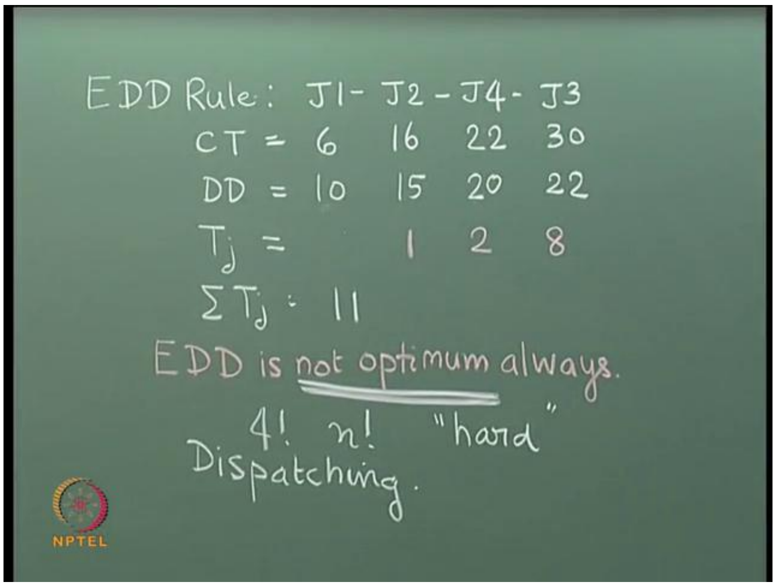
Now, we look at the third objective, which is minimize sigma T j, we have already defined T j, T j is defined as the tardiness associated with the completion times and due dates, so tardiness T j of job j, so T j is maximum over 0 coma C j minus D j. Now, C j is the completion time or time, at which job j is completed, D j is the due date associated with job j, now if the job completes later than the due date, then it is tardy in such case C j minus D j will be positive.

So, maximum of 0 comma C j minus D j will be positive and tardiness will be positive, if on the other hand, the job completes ahead of the due date, then C j minus D j will be negative, because C j the completion time will be smaller than the due date. So, when the job is early, which means C j is smaller than D j, then C j minus D j becomes negative on the tardiness associated with job j becomes 0.

So, tardiness of job j is maximum of 0 comma C j minus D j and total tardiness is given here, now the tardiness has two terms, which is the completion time term, as well as the due date term. Since we want to minimize the total tardiness, we would like to begin with we would like to arrange or sequence the jobs, in non decreasing or increasing due dates and hope that such a sequence will turn out to be optimum.

So, first evaluate that, so the rule here that, we try to use, this called E D D, which is call earliest due date, where jobs are arranged in the order of increasing or non decreasing value of the due date. Now, let us apply the earliest due date rule to the example problem and first compute the total tardiness.

(Refer Slide Time: 17:35)



EDD Rule : J1 - J2 - J4 - J3

CT =	6	16	22	30
DD =	10	15	20	22
T_j =		1	2	8

$\sum T_j$: 11

EDD is not optimum always.

4! n! "hard"

Dispatching.

So, now, based on earliest due date rule, we first sort the jobs according to the due dates, so the dues dates given are 10 15 22 and 20, so job 1 comes first job 2 comes 2nd job 4 comes, 3rd and job 3 comes 4th. So, based on the earliest due date rule the order will be J first 1 J 2 2nd J 4 3rd and J 3 will be the 4th. Now, the moment, we find out this order the completion times are J 1 starts at time equal to 0 and finishes at times equal to 6, J 2 starts at time equal to 6 finishes at time equal to 16, J 4 starts at 16 and finishes at 22 and J 3 finishes at 30 the last of the jobs always finishes at the Makespan.

Now, due dates that we have here, so the due dates are for job 1 the due date is 10, so due date is 10 15 20 and 22, now let us find out the tardiness associated with job j, now here, the completion time is 6, the due date is 10, therefore it is early. Due date is 15 completion time is 16, so tardiness is 1, due date is 20 completion time is 22, so tardiness is 2 and due date is 22 completion time is 30, so tardiness is 8, so sigma T j for this example is 11, so total tardiness is 11.

So, it is very customary to use the earliest due date rule to try and minimize the total tardiness, for example if we had used, we showed some calculation here. So, when we use the rule J 1 J 2 J 3 and J 4, we found that the total tardiness was 13, the earliest due date rule gives us a total tardiness equal to 11. Now, only difficulty here is that while, here we can say confidently that the shortest processing time will always give the minimum value of sigma C T or M C T we cannot say that E D D rule is optimal to minimize the total tardiness, so E D D is not optimum always.

Now, there are examples, through which we can show that the E D D rule is not always optimum, though in our example, we have calculated a total tardiness of 11, which actually turns out to be optimum for this particular illustration. But, E D D need not give us the optimum solution always, then we get into the next question of, can we get to the optimum solution always and if, so how do we get to the optimum solution always.

Because, the 2 earlier objectives of Makespan and total completion time, we have said that for the Makespan any sequence is optimum, so I could possibly write here and say any sequence or every one of the sequences is optimal, which will gives us a Makespan of 30. As far as minimizing sum of completion times, we could say that shortest processing time rule minimizes sum of completion times, earliest due date rule tries to minimize the total tardiness, but it need not give us the optimum solution always.

Now, in this case, if we need a method to find out the optimum solution always then that method would either explicitly or implicitly, it has to evaluate all the 4 factorial or in general n factorial sequences that are possible and then only it can gives an optimum solution. Sometimes, it evaluates explicitly, sometimes, it evaluates implicitly, because n factorial becomes very large, when n becomes large for example, if we look at n equal to 20, the number of computations is extremely large, when we want to enumerate n factorial.

So, we would look at methods that would implicitly evaluates a part of n factorial, but tell us there, it has a evaluated that and then by evaluating fewer than n factorial, it could give us the optimum solution. So, several branch and bound algorithms, try to do that by implicitly evaluating some of the solutions, but n factorial is very large and these sequencing and scheduling problems become hard problems.

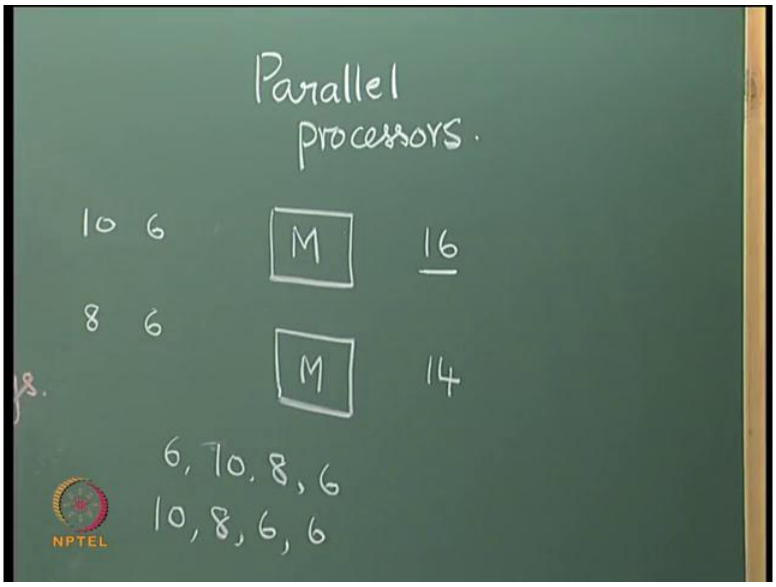
Particularly, when the problem size increases, when we, so far we have not found an algorithm that runs in other than n factorial explicitly or implicitly, because the computational effort to find out n factorial becomes exponential and very large as n increases. Therefore in these class of problems, it is customary to look at thumb rules or what are called dispatching rules, to try and get the best solution.

Earliest due date is one such popular dispatching rule, though earliest due date does not guarantee the optimum solution, in the case of the single machine tardiness minimization. Now, we have seen three objectives with respect to single machine problems and at the end of it two very important rules have come out, one is called the S P T rule the shortest processing time rule, the other is called E D D rule the earliest due date rule, now these 2 rules are also going to be used in later scheduling problems, such as job shop scheduling.

While the S P T provides an optimum solution, to the problem of minimizing sum of completion times, earliest due date is an extremely popular rule to address problem that have due date related objectives or due date related measures, E D D does not guarantee the optimum solution. Now, there are a couple of other things, which we would look at for example, instead of minimizing total tardiness, if we try to minimize total lateness, where lateness is simply L j is C j minis D j, if we want to do that, then we could realize that, if we want to minimize sigma C j minus D j.

D j's are known, so it is enough to minimize sigma C j and we already know that S P T minimizes sigma C j, so S P T also minimizes mean lateness or total lateness. And there are other rules, which would say that, if S P T or E D D gives only one tardy job then it minimizes the number of tardy jobs and so on. So, there are some interesting rules and algorithms, which are available, but then we have seen a small sub set of it and at the end of it I would say here that, at the our discussion on single machine problems. We realize the two important rules have come out, one is called the S P T rule, the other is called the E D D rule or the earliest due date rule, now we spend a few minutes on minimizing in parallel processors.

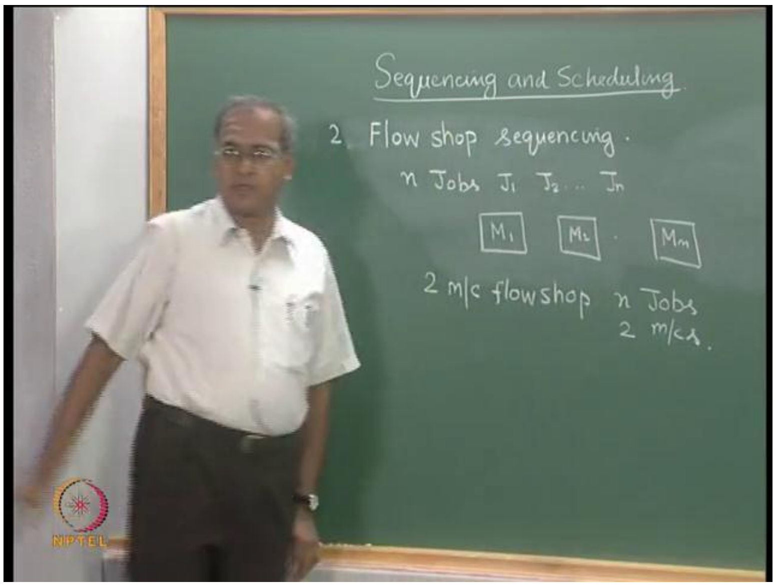
(Refer Slide Time: 26:46)



On parallel processors, now if we assume that, there are two machines instead of 1 and they are identical. Now if we want to minimize the Makespan, we have 4 jobs with 6 10 8 and 6, so with 6 10 8 and 6. Here again, one can show that, the problem of minimizing Makespan becomes hard and difficult a simple rule that is often used is L P T, which means arrange them in the non decreasing order of non increasing order or decreasing order of processing times, to get 10 8 6 and 6 and starting locating them to the 2 machines this way.

So, 10 goes here 8 comes here, the next 6 will come here and this 6 will come here, if there is a 5th job, it will go here, the 6th 7th 8th and so on, now here the Makespan is equal to 16, here the Makespan is 14. So, the actual time, at which all the jobs are over is the maximum of these 2, which happens to be 16, so one would say that, if we apply this rule then the Makespan that, we get is 16. So, there are other rules and algorithms, to try and minimize other objective, such as mean completion time tardiness and so on. But, has I have mentioned the end of discussion on single machine problems, we learnt two important rules, one is called the shortage processing time rule and the other is called the earliest due date rule.

(Refer Slide Time: 29:18)

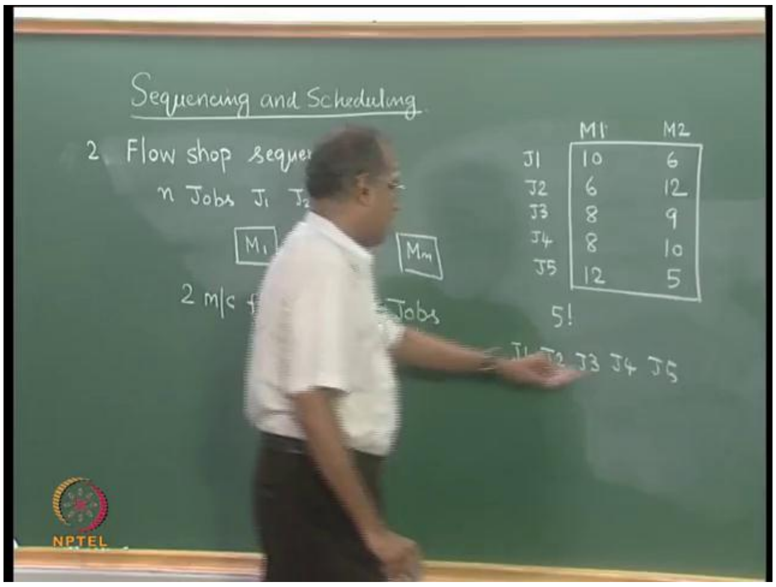


So, now we get into the next set of problems and sequencing and scheduling, which is called flow shop sequencing called flow shop sequencing, we explain it through a small example. As we have already mentioned, a flow shop is a shop, where say we have n jobs J 1 up to J n and then we have M machines, which we would call as M 1 M 2 and M m an important condition in a flow shop is that, all the jobs will have to visit all the machines, in the same sequence.

So, J 1 requires processing on M 1 M 2 M 3 up to M m, it also requires processing in the same order except that, the processing times are different - the same with J 2, same with J 3 and so on. So, we it is customary to say that the machines are already arranged, in the order, in which all the jobs visit them, so we could say, we would call the machine, which all the jobs visit first as M 1, a machine which all the jobs visit second as M 2 and so on.

So, the machines are already arranged in the order of visit of the jobs, so this is a typical flow shop problem and then we try to look at, what we do for objectives such as Makespan mean completion time and so on. In this lecture series will concentrate only on the Makespan for flow shop sequencing, now we explain it through a small example, we first address, what is called the 2 machine flow shop problem, where there are n jobs and there are only 2 machines.

(Refer Slide Time: 31:42)

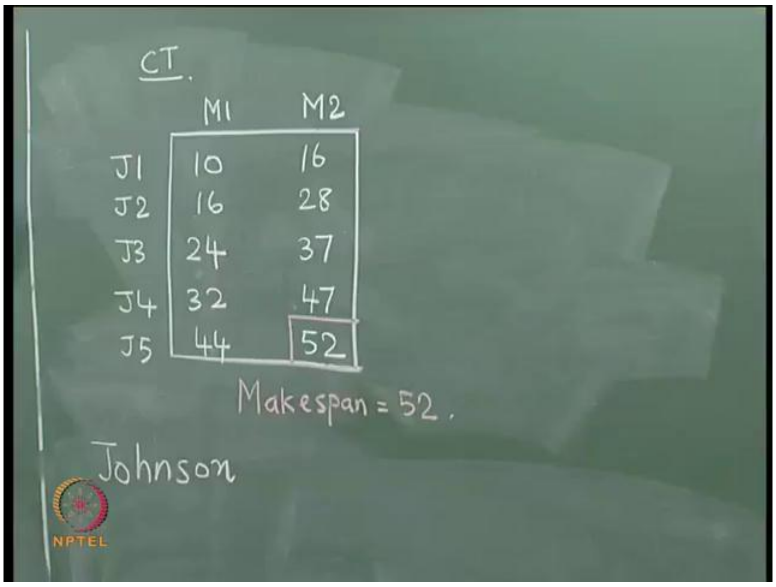


	M1	M2
J1	10	6
J2	6	12
J3	8	9
J4	8	10
J5	12	5

So, we take a small example, to explain the n job 2 machine problem, now this is an example, where there are 5 jobs, which we call J 1 J 2 J 3 J 4 and J 5 and there are 2 machines M 1 and M 2. Now, we want to find out, the order or sequence, which minimizes the Makespan, so there are 5 jobs, so there are maximum of 5 factorial sequences, in which we can arrange these 5 jobs. Now, we will check about this 5 factorial after a while, but then we right now assume that, there are 5 jobs and they can be arranged in 5 factorial ways.

And we right, now assume that one of the 5 factorial ways, by which we can send the jobs is going to be optimal, for the Makespan. So, first let us try and compute the Makespan for a given sequence and then we try and optimize the sequence, so to make it very easy, we try and compute the Makespan, for the sequence J 1, J 2, J 3, J 4 and J 5.

(Refer Slide Time: 33:26)



CT.

	M1	M2
J1	10	16
J2	16	28
J3	24	37
J4	32	47
J5	44	$\boxed{52}$

Makespan = 52.

Johnson

Now, first for a given sequence, let us find out the completion times, now these are completion times on machines M 1 and M 2, so we first send job J 1, so job J 1 goes first. So, machine both the machines are free and their available at time equal to 0, so J 1 requires 10 units of time on M 1, so it will start at 0 and comes out time equal to 10 remember that, these are completion times. So, it starts at 0 and comes out at time equal to 10, so at time equal to 10, it goes to M 2, M 2 is free, at M 2 it requires another 6 units, so it comes out at time equal to 16.

Now at 10 M 1 is free and therefore, the next job goes to M 1, so right now we are evaluating it for a sequence J 2, so J 2 comes in next, machine M 1 is available at time equal to 10, therefore J 2 starts at time equal to 10 and finishes at time equal to 16. Now, in this case exactly at time equal to 16, J 2 finishes at M 1, now M 2 is also free at time equal to 16, so J 2 straight way takes M 2 at time equal to 16, requires another 12 units of time, so at time equal to 28, J 2 finishes at M 2 and J 2 goes out of the system.

Now, once again at time equal to 16 M 1 is free. So, M 1 can takes its next job, which happens to be J 3, because we are evaluating a sequence 1 2 3 4 5, so it starts at 16 and finishes at time equal to 24, 16 it requires a further 8, so 16 plus 8 24 8 comes out. Now, at time t equal to 24, J 3 comes out of M 1 and J 3 wants use M 2, but M 2 is free only at time equal to 28, so J 3 has to wait for 4 units of time before, it could go to M 2.

So, right now we assume that, there is enough space and buffer between M 1 and M 2, so that the person, who is working can take J 3 out of M 1 keep it in the buffer space and then take up the next job. So, the next job can enter at time equal to 24 into M 1, but to finish this computation, now we say that the J 3 can take M 2 only at time equal to 28, so at 28, it takes on it goes to M 2, so 28 plus 9 is 37. Now J 4 can enter at 24, because at 24 it is over, the person has taken the job and kept it outside M 1 is free.

So, J 4 starts at time equal to 24 takes a further 8, here finishes at time equal to 32, now J 4 again has to wait for 5 more units, so that M 2 becomes free at 37 and then it goes to M 2 at 37 takes 10 more units finishes at 47 and then it comes out. Now, at 32 J 5 can enter M 1, so J 5 enters M 1 at time equal to 32, once again, we assume that, there is enough space in between for the person to take it and keep it, so that M 1 can be released for it is next job, so it starts at 32 and finishes at 44 - 32 plus 12 is 44.

And then it has to wait for 3 more time unit to get into M 2, starts at 47 finishes at 47 plus 5, 52 and time equal to 52 all the jobs are over and therefore, for this given sequence the Makespan is 52. Now, the question is what is the sequence or what is the order, in which we can send this, such that the Makespan is minimized? We also realize something interesting from this particular table. If we see very carefully 44, which is the time, at which M 1 completes all the jobs is actually, the sum of the processing times here 10 16 24 32 plus 12, 44.

So, M 1 is not idle, till it finishes everything then M 1 becomes idle in this case for 8 more time units or 8 more minutes, other thing that, we observe is M 2 cannot start at time equal to 0, because of the flow shop condition. So, M 2 can now start here at 10 and then we also realize that, there were some delays that happened here, so there was a delay of 4 units here, there is a delay of 5 units here, there is a delays of three units here, so 4 plus 5 9 plus 3 12.

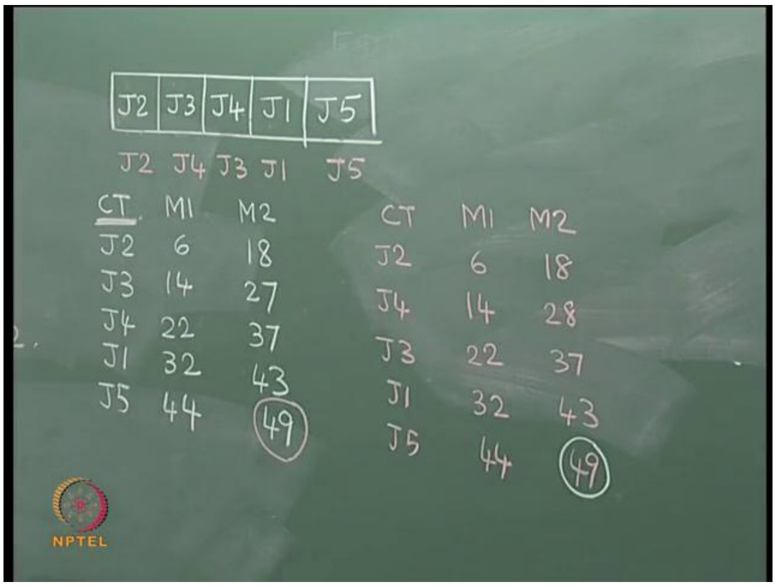
Now, we look at this 18 27 37 plus 5, 42 plus the other 10 was 52. 18 plus 9 27 37 42 10 52, but then it is started at 10, 10 plus 42 is 52, M 2 also was not idle. Now sequencing and scheduling problems become important. Because, this number, at which it completes the Makespan largely depends on a couple of things. It depends on 2 very important things here, which is called job waiting for the machine and machine waiting for the job.

Now, in this case, we had situations, where the job was waiting for the machine, this job was over at time equal to 24, but then the job had to wait till 28, here the job was over at 32, it had to wait till 37. Now, you look at this case machine is waiting for a job the machine was available at 0, but then the machine has to wait for this job it has to wait for 10 minutes. So, it is usually the delays that happen, when the job has to wait for the machine and the machine has to wait, for the job that creates, this kind of an issue, where the objective changes.

So, the problem that we are looking at is given a sequence, we can find out the Makespan, what is the sequence that minimizes the Makespan? Now, a very popular algorithm is called the Johnson's algorithm was developed by Johnson in the year 1954, which tries to minimize the Makespan on 2 machines. Now, Johnson not only gave a very simple algorithm, but he actually gave a very elegant proof for the algorithm.

So, we will see the algorithm alone in this lecture series, we will not see the proof, but then we see how intuitive Johnson's algorithm is when, it comes to solving a 2 machine flow shop sequencing problem. So, Johnson's algorithm works like this, a very simple way to do it is there are 5 jobs.

(Refer Slide Time: 41:08)



J2	J3	J4	J1	J5

J2 J4 J3 J1 J5

CT.	M1	M2
J2	6	18
J3	14	27
J4	22	37
J1	32	43
J5	44	49

CT	M1	M2
J2	6	18
J4	14	28
J3	22	37
J1	32	43
J5	44	49

So, create a small table with 5 positions, now find out the smallest number in this table, the smallest number is 5 and 5 happens to be for job 5 on machine M 2, so we look at this and the rule says, if the smallest number is on M 1, then come from the left and if the smallest number is on M 2 come from the right. So, smallest number, which is 5 is on M 2, so we come from the right in this table and since that happens, for J 5, now we come from the right and write J 5 here as the last job.

So, once again look at this smallest number, the smallest number is 5 and check whether, it is on M 1 or whether, it is on M 2 and for, which job it is, so in the case the smallest number is on M 2 and it is for J 5. Since it is on M 2 come from the right and write that job in the first available position from the right, so first available position from the right is here, so J 5 goes here, now remove this job from the list.

Now, look at the rest of the table and try to find out the smallest number, the next smallest number is 6 and it happens to be on M 2 for J 1 and also happens to be on M 1 for J 2. So, when we have a situation like this, where the smallest number is coming on M 2 as well as M 1, we may be tempted to believe right now that there is a tie, because 6 is appearing in 2 places and 1 case, it is on M 2, other case it is on M 1.

The fact that, it is on M 2 in one case and on M 1 in the other indicates that, it is actually not a tie and we can actually break it, arbitrarily there are no problem at all, if it happens to be on M 2 as well as on M 1 for 2 different jobs. So, we could take either this or we could take this, now let us take J 2 on M 1 as the smallest number, so J 2 on M 1 means, since it is on M 1 the rule says come from the left and take the first available position from the left. So, it is on J 2 on M 1, because it is M 1 come from the left, take the first available position, so from the left take the first available position and put J 2.

Now, remove J 2 from the table temporarily, now once again find the smallest number has to be is 6, it is on M 2 and it is for J 1, since it is on M 2 come from the right and take the first available position, so from the right you take the first available position and put J 1 here. Now, at this point we realize, what we said earlier, now we had the minimum coming here, for J 1 on M 2 and the minimum coming here, which is J 2 on M 1, we first took this and we put J 2 here and then we ended up writing J 1 here.

If we are taken it the other way, we would have taken this first then we would have written J 1 here and then J 2 would have come here, the sequence has not changed, because the smallest number happened to be on M 1 for a particular job and on M 2 for some other job. So, now, remove J 1 also, so 3 jobs have been put in the table, 3 jobs have gone out, so what we have remaining are J 3 and J 4, now find the smallest number, which happens to be 8, which is on M 1, which I for J 3 as well as, it is for J 4.

Now here there is a tie whether, we take this J 3 on M 1 or whether we take J 4 on M 1, so as long as the smallest number happens to be on 2 different machines, there are only 2 machines, happen to be on 2 different machines, there is no tie, the sequence does not change or the sequence is not affected. But, if there is a tie like this, when the smallest number happens to be on M 1, now we would either take J 3 or take J 4, so first let us take J 3.

So, since it is on M 1 come from the left and take the first available position, so we take J 3 here, now remove this J 3 there is only 1 job remaining and there is only 1 position vacant, so put J 4 here. Now, we had a tie, we had a tie between J 3 and J 4, both of them have a smallest number and it was on M 1, we resolve the tie by taking J 3 first, now what would have happen, if we had taken J 4 first, if we had taken J 4 first we would have got the sequence J 2 J 4 J 3 J 1 and J 5.

So, Johnson's algorithm in this for this particular example gives 2 sequences, now 2 sequences happened, because of a tie occurring here and we had one sequence, which we resolved by putting, J 3 first and then J 4 and the other sequence by putting J 4 first and then J 3. Now, let us try and evaluate the Makespan, for both the sequences will first do that for this sequence, so the sequence that we are taking is J 2 J 3 J 4 J 1 and J 5 and completion times on M 1 and M 2.

So, J 2 goes first J 2 finishes starts at 0 finishes at 6, now J 2 goes to M 2 at 6 takes another 12 units, so finishes at 18. J 3 goes next J 3 can start on M 1 at 6 needs another 8 units of time finishes at 14, but M 2 is available only at 18. So it takes another 9, 18 plus 9, 27. Now, J 4 can start at 14 on M 1 takes another 8 units, so 14 plus 8, 22. Now it has to wait till 27, for M 2 to be free takes at M 2 at 27, takes another 10 units of time finishes at 37.

Now J 1 can starts at 22 takes another 10 units of time finishes at 32, 22 plus 10. It has to wait for another 5 units for 37, so takes M 2 at 37 plus 6, 43 it comes out. J 5 is the last job, it can start at time equal to 32 takes another 12 units of time, so finishes at time equal to 44. And now, J 5 comes out at time equal to 44 M 2 is already free at 43, but J 5 can take M 2 only at time equal to 44, takes another 5 units finishes at 49 and now all jobs are over at time equal to 49, which is the optimum Makespan according to Johnson's algorithm.

Now here we can see both, we can a situation, where this job is coming out at 14, but the machine is available at 18 so here is the case, where the job is waiting for the machine to be free. Here the job comes out at 44, but the machine is already free at 43, but the machine has to wait for the job to come, so there is a machine waiting time of, one which again increases the Makespan.

So, the Makespan increases due to two things one is job waiting for the machine the other is machine is waiting for the job. It is only these 2 delays that makes sequencing and scheduling problems important and scheduling and sequencing problems difficult. So, based on the Johnson's algorithm, where we got the sequence, we got the Makespan to be 49, but we also got another sequence here, so we will quickly try and compute the Makespan for the other sequence.

So, here we are again computing cycle times, we have M 1 completion times, we have M 1 M 2, the sequence is J 2 J 4 J 3 J 1 and J 5, now for J 2, it is the same, so it is 6 and 18. Next we have J 4, so J 4 comes here. So, J 4 takes M 1 at 6 finishes at 14 has to wait for 4 units of time, so goes to M 2 at 18 and finishes at 28. Now, J 3 can start at 14 takes another eight units of time, so 14 plus 8, 22, waits for 6 more units goes to M 2 at 28 and then plus another 9 finishes at 37.

Now J 1 can go to M 1 at 22 takes 10 units of time, finishes at 32 waits for 5 more units of time to go to M 2 at 37 takes another 6 units and comes out at 43. Now J 5 goes to M 1 at time equal to 32 takes 12 more units finishes at 44. Now, M 2 is waiting, so it goes to M 2 at 44 takes another 5 units of time and finishes at 49. So this sequence also gives us the Makespan of 49. So Johnson's sequences, because of tie is if we have more than one sequence all the sequences are optimal and all of them give the same amount of Makespan.

Now, Johnson's algorithm gives the optimum Makespan, for a general m jobs, but 2 machine flow shop and Johnson's algorithm is based on a very simple rule, that if you have n jobs create a table with n slots, find out the smallest number in the table. Now, if this smallest number is on M 1, then come from the left and put the corresponding job in the first available position, if the smallest number happens to be on M 2, then come from the right and put the corresponding job in the first available position.

Once a job is entered into the table temporarily eliminate or remove the job and once again find the smallest number, repeat this till all jobs go into this, if there is a tie, can be broken arbitrarily. But, ties can result in more than one sequence a tie will happen, if the smallest number is on a particular machine say M 1 and more than 1 job has that. Similarly, tie will happen when, it is on M 2 and more than one job has it, but if the same smallest time happens to be on M 1 and M 2, for 2 different jobs then it is not a time.

So, get as many solutions as there are ties all of them are optimum with the same value of Makespan, now Johnson's algorithm is seen is the starting point of research in sequencing and scheduling. Because, what appears to be a complicated problem with the worst case possibility of n factorial Johnson came off with the very, very elegant algorithm, which could solve the 2 machine case.

Now Johnson's algorithm triggered a whole lot of research in sequencing and scheduling and for well over 55 plus years 100 and 1000s of people have been working on sequencing and scheduling problems. How, Johnson's algorithm can actually be extended, to a very special case of the 3 machine problems, now Johnson's extension to the three machine, we will see in the next lecture.