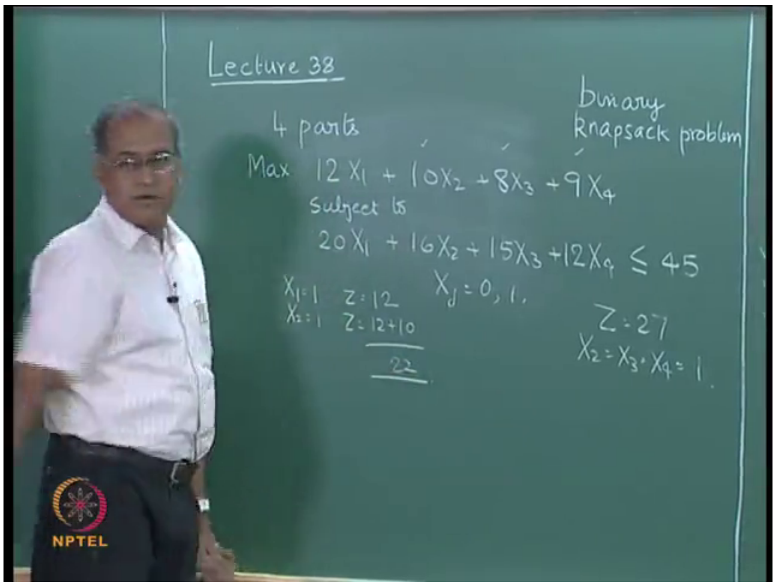
Manufacturing Systems Management
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras

Lecture - 38
FMS Loading problem

This lecture we continue the Branch and Bound Algorithm. The previous lecture we were looking at the part selection problem which is about; which are the parts that have to be selected to be manufactured in an FMS.

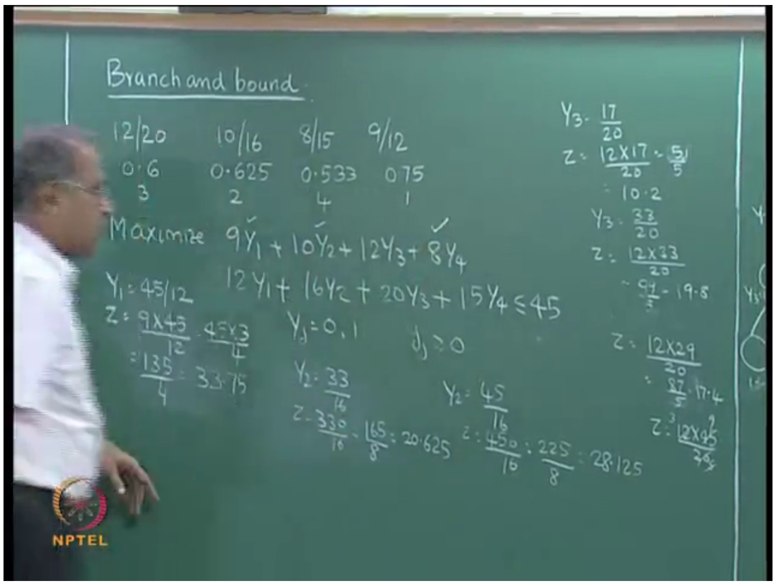
(Refer Slide Time: 00:27)



So, we looked at this example where there are 4 candidate parts giving us a profit of 12, 10, 8, and 9 and requiring times 20, 16, 15, and 12 and assuming that 45 units of time is available on the FMS.

So, we formulated this binary problem and then we rearrange the variables or rename the variables based on the ratio of the profit to the time consumed.

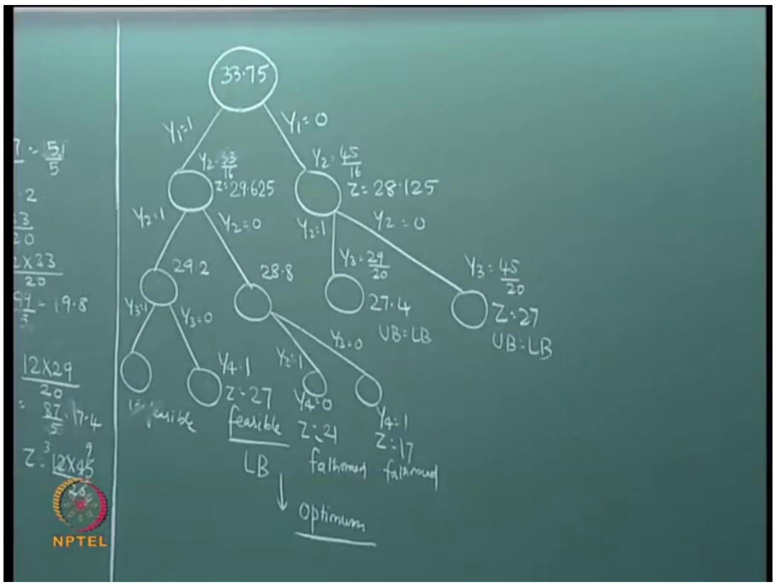
(Refer Slide Time: 01:01)


Branch and bound
12/20 10/16 8/15 9/12
0.6 0.625 0.533 0.75
3 2 4 1
Maximize 9Y1 + 10Y2 + 12Y3 + 8Y4
12Y1 + 16Y2 + 20Y3 + 15Y4 ≤ 45
Yj = 0, 1
Y1 = 45/12
Z = 9×45/12 = 45×3/4 = 135/4 = 33.75
Y2 = 33/16
Z = 330/16 = 165/8 = 20.625
Y2 = 45/16
Z = 450/16 = 225/8 = 28.125
Y3 = 17/20
Z = 12×17/20 = 51/5 = 10.2
Y3 = 33/20
Z = 12×33/20 = 99/5 = 19.8
Z = 12×29/20 = 87/5 = 17.4

And the problem was rewritten by introducing the Y variables instead of the X variables such that the variables are written in the non-increasing order of the coefficients.

Now, this would help us in getting the L B optimum because the leftmost variable will be the variable in the solution if we solved an L B. Then we started the branch and bound algorithm and we reached a point where we had a feasible solution with Z equal to 27.

(Refer Slide Time: 01:35)



So, profit of 27 is possible it is a maximization problem. So, feasible solution will give us a lower bound on the optimum which means the optimum is 27 or more and in each node we solve an L B. So, L B is an upper bound for a maximization problem. So, we stopped here where we said there we have to proceed from this node because this has an upper bound of 28.1 to 5 and the best value so far is 27, we can get a 28 if it exists. So, we should try and get that 28; 28 comes because all these objective function values are integers and therefore, the optimum solution will be an integer. So, 28.1 2 5 is effectively 28.

So, now we proceed here with Y 2 taking a value 0 or 1. So, this is Y 2 equal to 1 and this is Y 2 equal to 0. So, this gives us 2 notes this is Y 1 equal to 0 Y 2 equal to 1 both of these are fixed. So, Y 1 equal to 0 Y 2 equal to 1 means we use 16 units here. So, 20 9 units are available. So, Y 3 will be 29 by 20. So, Y 3 will be 29 by 20 and the objective function value will be 12 into 29 by 23 into 29 is 87 by 20, 87 by 5 17 fives or 85 17.4 plus this has Y 2 equal to 1 so 17.4 plus 10. So this gives us a bound of 27.4.

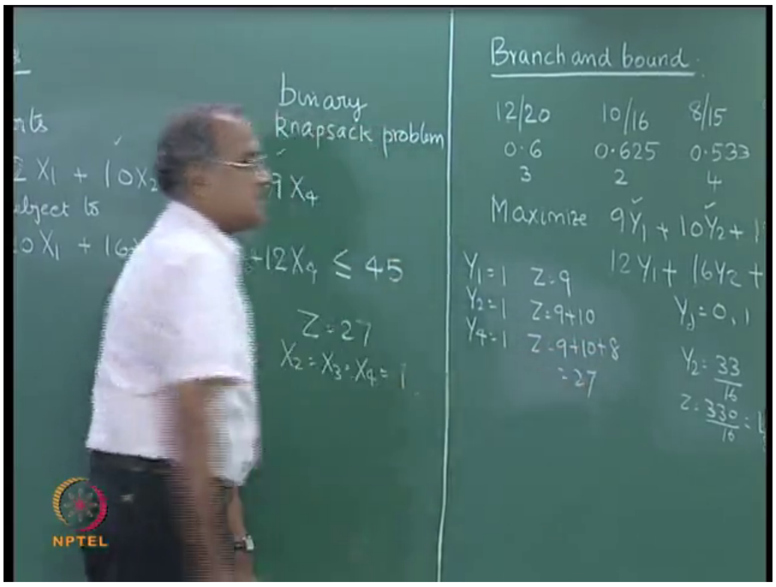
So, now we use the condition since it is a binary problem and all these objective function coefficients are integers. So, at the optimum the objective function value has to be an integer. So, if we proceed from here we can only get the lower integer value of this or less we could get 27 or less we already have a solution here with 27. So, we do not need to proceed from this. So, this is fathom because upper bound is equal to lower bound, upper bound is 27, lower integer value of 27.4. So, upper bound is the equal to the best lower bound. So, even if we proceed from here we can get only 27 or less. So, we do not proceed.

Now, the other 1 is Y 1 equal to 0 Y 2 equal to 0 which means both are kept at 0 45 units available. So, Y 3 equal to 45 by 20. So, Y 3 equal to 45 by 20 and objective function value is 12 into 45 by 20. So, this is 3 by 59. So, Z equal to 27. So, once again from here upper bound is 27 proceeding we can only get solutions with 27 or less. So, we do not proceed. So, we fathom here by saying upper bound is equal to lower bound. So, we do not have any more notes to proceed. So, this is the optimum solution. So, this gives us a solution with this is Y 1 equal to 1 Y 2 equal to 1 Y 4 equal to 1. So, Y 1 equal to 1 Y 2 equal to 1 Y 4 equal to 1 with profit equal to 27 and 12 plus 16 28 plus 15 43 units of time are taken and 2 units of time is not used this is the same solution that we got when we solved it by integer programming.

Now, 1 of the reasons you look at a branch and bound is 1 is the time taken as such by a branch and bound could be little less than the time taken to solve this using a solver and in the event we do not have a solver then we could use this branch and bound algorithm and solve it branch and bound algorithm uses principles of lower bound upper bound relationship between linear programming and binary I P and so on. So, branch and bound also gives us the optimum solution and branch and bound to solve knapsack problems are extremely popular and 1 can easily write a branch and bound code to solve this well. Now if we do not want to try and get optimal solutions and if we are looking only at heuristic solutions then let us see what happens?

Now, let us look at this 1 and try to get some heuristic solutions. So, this is the formulation. So, we simply take these variables as they appear and see whether we can make them. So, Y 1 is the variable or the product that actually has the highest profit their profit per unit time even though in terms of highest profit 12 is the highest, but it requires 20 units of time. So, profit per unit time the highest ratio happens to be for the variable Y 1. So, we start from the left and say Y 1 equal to 1 Z equal to 9.

(Refer Slide Time: 07:42)



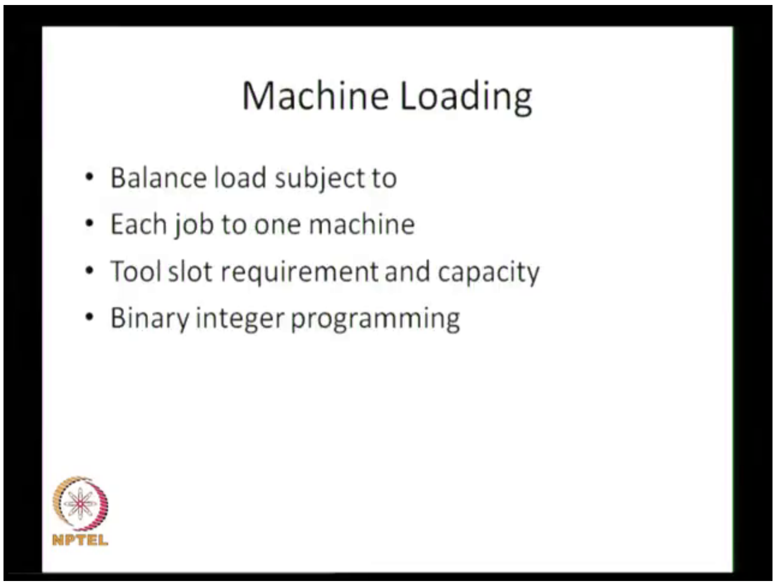
Now, when we put Y 1 equal to 1 12 units are taken away. So, 33 units are available Y 2 requires 16 units. So, Y 2 is equal to 1 Z is equal to 9 plus 10 19. So, Y 1 equal to 1 Y 2 equal to 1 would take 28 units. So, 17 units are available. Now we look at Y 3 equal to 1 Y 3 requires 20. So, Y 3 we do not produce Y 3 equal to 0 and then we go back and check for the 17 what happens to Y 4 Y 4 requires only 15. So, this gives us Y 4 equal to 1 with Z equal to 9 plus 10 plus 8 which is 27, which is the same optimum solution for this particular instance, but if we had not sorted the variables and if we had tried to solve this problem as it is without sorting or rearranging the variables and if we apply this heuristic.

Now, we would start with X 1 equal to 1 Z equal to 12 now X 1 equal to 1 takes away 20 units of time. So, 25 is available. So, this would given give us X 2 equal to 1 with Z equal to 12 plus 10. So, we have used up 36 units and we have 9 units. So, X 3 we cannot do X 4 also we cannot do. And therefore, we end up getting a solution with 22 as against 27. So, if we apply this heuristic it is very necessary that we sort the variables according to the coefficient and then we choose the variables in the order of non-decreasing value of this coefficient.

Now, we should also bear in mind that this after sorting it need not give us the optimum solution all the time, it gave us this heuristic gave us the optimum solution for this particular example, but if there is another example where the optimum solution happens to be say 1 3 and 5 if in 5 variables this heuristic sometimes can give us 1 2 and 4 which is lesser in terms of objective function value than 1 3 and 5. So, we could have situations we could have situations where 1 3 and 4 would be optimal whereas, the heuristic would show 1 2 now 1 3 and 5 1 3 and 5 could be optimal, whereas the heuristic could show or 1 3 and 4 could be optimal, whereas a heuristic could show 1 2 and 5. So, we could get into situations like this.

So, this heuristic need not give us the optimum solution all the time, but nevertheless it is a very effective heuristic particularly when we take into account the ratios and the order of the variables based on that. So, this is how the part selection problem is solved in flexible manufacturing systems.

(Refer Slide Time: 11:16)

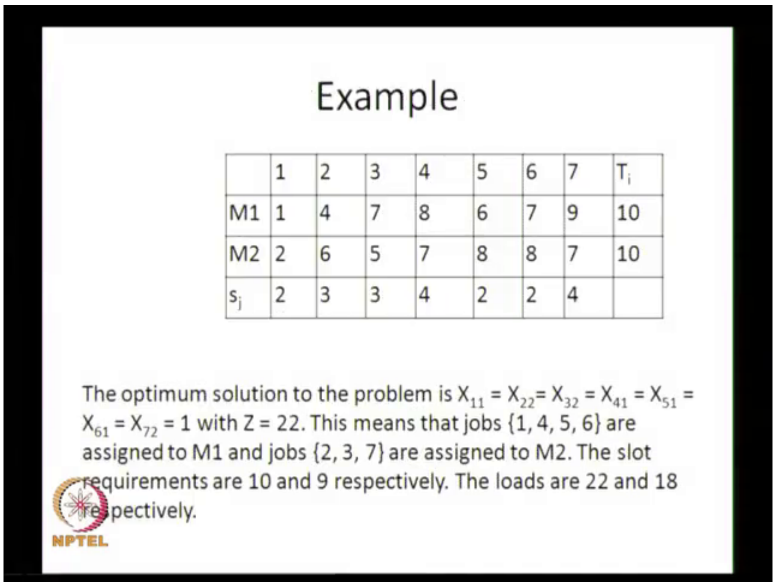


Machine Loading

- Balance load subject to
- Each job to one machine
- Tool slot requirement and capacity
- Binary integer programming

We now move to the second problem which is called the machine loading problem which talks about balancing the workloads subject to allocating each job to only 1 machine, there is a tool slot requirement and capacity. So, we will explain this problem through an example.

(Refer Slide Time: 11:37)



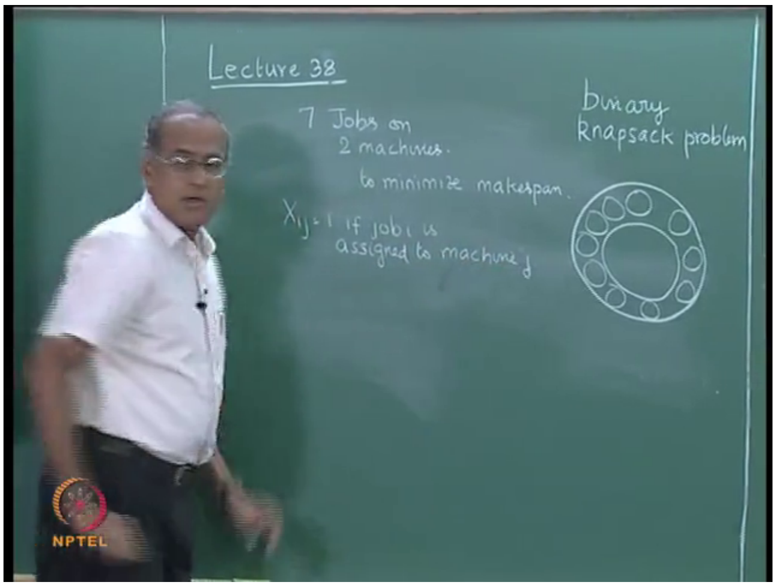
Example

	1	2	3	4	5	6	7	T_i
M1	1	4	7	8	6	7	9	10
M2	2	6	5	7	8	8	7	10
s_j	2	3	3	4	2	2	4	

The optimum solution to the problem is $X_{11} = X_{22} = X_{32} = X_{41} = X_{51} = X_{61} = X_{72} = 1$ with Z = 22. This means that jobs {1, 4, 5, 6} are assigned to M1 and jobs {2, 3, 7} are assigned to M2. The slot requirements are 10 and 9 respectively. The loads are 22 and 18 respectively.

Now, this is this example is for the machine loading problem, now here we assume that we have done the machine grouping problem which was 1 are the production planning problems listed. So, the machines are grouped together and there are 2 machines, now there are 2 machines and we have to load 7 jobs on 2 machines 7 jobs on 2 machines.

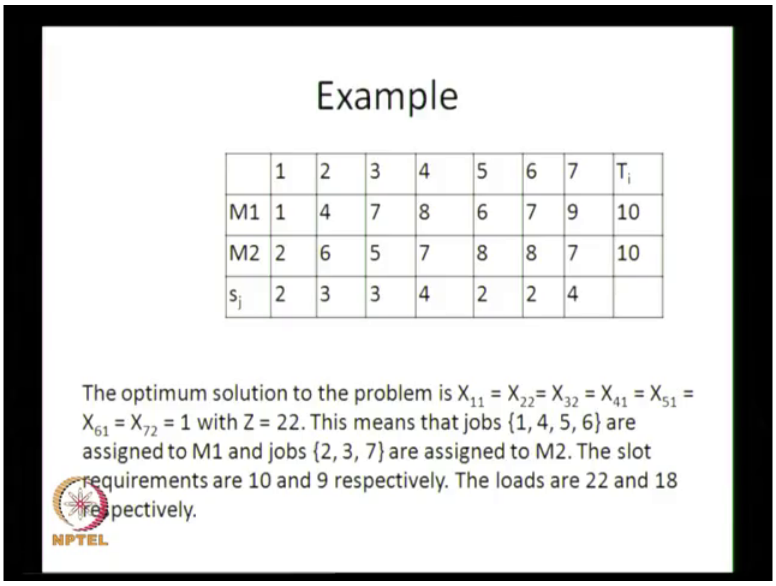
(Refer Slide Time: 12:05)



Now, this the processing times are given there is also an additional tool restriction which we will consider, now if we do not have this tool restriction then the problem is simply 1 of allocating 7 jobs on 2 machines to minimize makespan to minimize makespan. This comes under the category of non-identical parallel processors n jobs single machine, but non identical parallel processor which means 7 jobs on a single type of machine, but 2 machines are available processing times are different on different machines identical parallel processor means the processing times are the same on both the machines here you can see that job number 1 takes 1 unit of time if it is done on M 1 and 2 units of time if it is done on M 2.

But now we also have tool requirements which are there now we have to say that each machine M 1 has a tool drum with capacity of 10 tool slots. Typically the side view of a tool drum would look like this there will be a tool drum like this with slots here. So, we would assume that there are 10 slots 10 slots where tools are inserted. Now each job would require certain number of tools. For example, I can allocate 1 2 and 3 to the first machine.

(Refer Slide Time: 14:28)



Example

	1	2	3	4	5	6	7	T_i
M1	1	4	7	8	6	7	9	10
M2	2	6	5	7	8	8	7	10
s_j	2	3	3	4	2	2	4	

The optimum solution to the problem is $X_{11} = X_{22} = X_{32} = X_{41} = X_{51} = X_{61} = X_{72} = 1$ with $Z = 22$. This means that jobs {1, 4, 5, 6} are assigned to M1 and jobs {2, 3, 7} are assigned to M2. The slot requirements are 10 and 9 respectively. The loads are 22 and 18 respectively.

Because the tools required is 8 2 plus 3 plus 3, I cannot do 1 2 3 and 4 to the first machine because the tools required is 12 it can accommodate only a maximum of 10 tools it can accommodate.

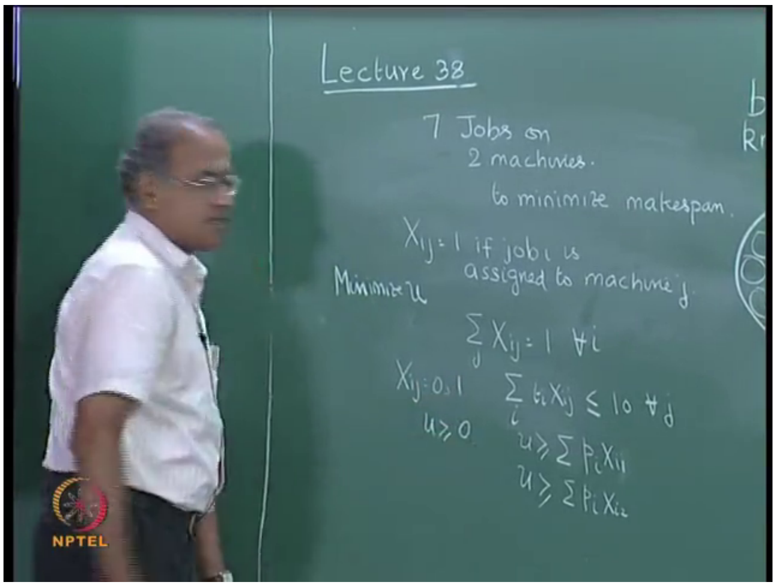
Now, the load balancing problem is also a problem which minimizes they makespan in some sense, when we try to balance the load on both the machines at the end what we are trying to do is as long as the balance results in a smaller amount of time on both the machines as smaller time on both the machines. Now all the 7 jobs are going to be done in parallel now suppose we allocate in as in this example suppose we allocate 1 2 3 and 5 to the first machine and we allocate 4 6 and 7 to the second machine suppose we do that, then on each machine the 10 tool slots are going to be occupy. So, it is feasible.

So, what we are going to assume is these jobs 1 2 3 and 5 are available they go to the first machine, 4 6 and 7 go to the second machine 1 after another, but when these 4 jobs are allocated to the first machine that 10 tools that are required are fixed on the tool drum.

So, when the work begins these 4 jobs are taken 1 at a time the tools are all available and in 1 batch they go to the first machine. Similarly 4 6 and 7 go as 1 batch to the second machine and all of them can come out once the entire processing is over, sometimes we may even have a situation that the job goes there are enough tools the work is carried out the job moves the next job comes and so on. So, each job is completed at certain point in time, but all the 4 jobs are completed on M 1 at a certain time and all the 3 jobs are completed on M 2 at a certain time and that time is what we are going to balance. So, that all 7 can be completed within a certain amount of time which is minimized

So, the problem will be like this. So, the problem will be like this there are 7 jobs and 2 machines. So, X i j equal to 1 if job i is assigned to machine j job i is assigned to machine j.

(Refer Slide Time: 17:49)



Lecture 38

7 Jobs on 2 machines. to minimize makespan.

$X_{ij} = 1$ if job i is assigned to machine j.

Minimize u

$\sum_j X_{ij} = 1 \ \forall i$

$X_{ij} = 0, 1$ $\quad \sum_i t_i X_{ij} \leq 10 \ \forall j$

$u \geq 0$ $\quad u \geq \sum p_i X_{i1}$

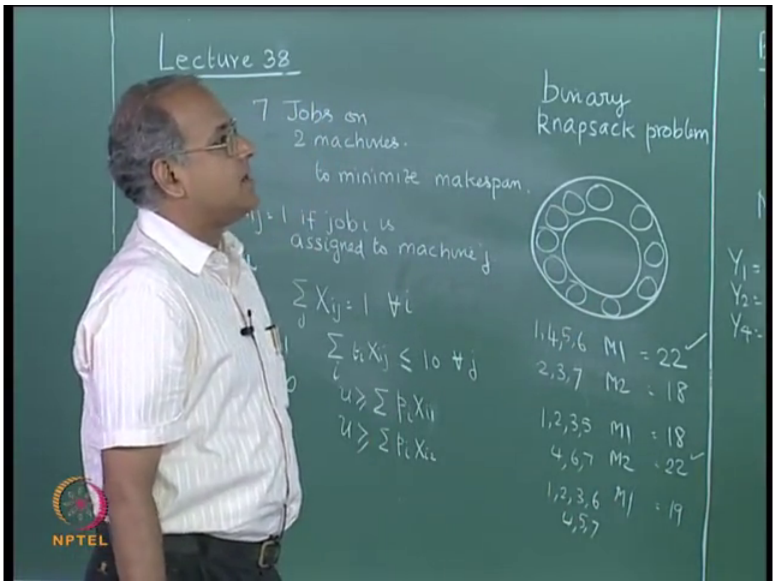
$u \geq \sum p_i X_{i2}$

Now, each job goes to only 1 machine. So, sigma X i j equal to 1 summed over j for every I, he job goes to only 1 machine the tool constraint has to be satisfied. So, sigma summed over i t i X i j is less than or equal to 10 for every j for each machine the number of tools required based on the jobs this less than or equal to 10.

Now, the load on the first machine will be sigma p i X i j or p i X i 1 and the load on the second machine will be sigma p i X i 2, now we want to balance the load. So, we want to minimize the maximum of these 2 loads. So, we will minimize u such that u is greater than are equal to this u is greater than are equal to this want to minimize the maximum of these. So, u is greater than or equal to all this. So, this will have 7 constraints that make sure that each job is assigned to 1 machine, this has 2 constraints to ensure that the tool requirement on each machine is met, this is a 1 constrained this is 1 constrained and the objective function is to minimize u.

So, we can solve this this is also a binary problem where X i j equal to 0 1, but there is an additional u greater than or equal to 0. So, it has 14 binary variables and 1 u which is either a continuous variable or an integer variable because processing times are all integers. So, when we actually solve this we get jobs 1, 4, 5, 6 are assigned to machine 1.

(Refer Slide Time: 20:25)



So, jobs 1, 4, 5, 6 are assigned to machine 1 jobs 2, 3, 8 or assigned to machine 2, 2, 3, 7 are assigned to machine 2.

Now, when we do this job 1, 4, 5, 6 on machine 1 has time 1 plus 8, 9, 9 plus 6, 15 plus 7 20 2 the tool required are 1, 4, 5, 6, 4 plus 2 6 plus 4, 10, 2, 3, 7 and machine 2 is 6 plus 5, 11, 11 plus 7, 18 in terms of tool requirement 2 3 and 7 is 3 plus 3 6 plus 4 10. So, the loads are 22 and 18 the most balanced load is 80, if we had looked at 1, 2, 3 and 5 on M 1 if we look at 1, 2, 3 and 5 on M 1, 4, 6, 7 on M 2 if we look at an alternate solution. So, 1, 2, 3 and 5 have tool equal to 10, 4, 6, 7 also have tool equal to 10. So, they satisfy all these constraints these 2.

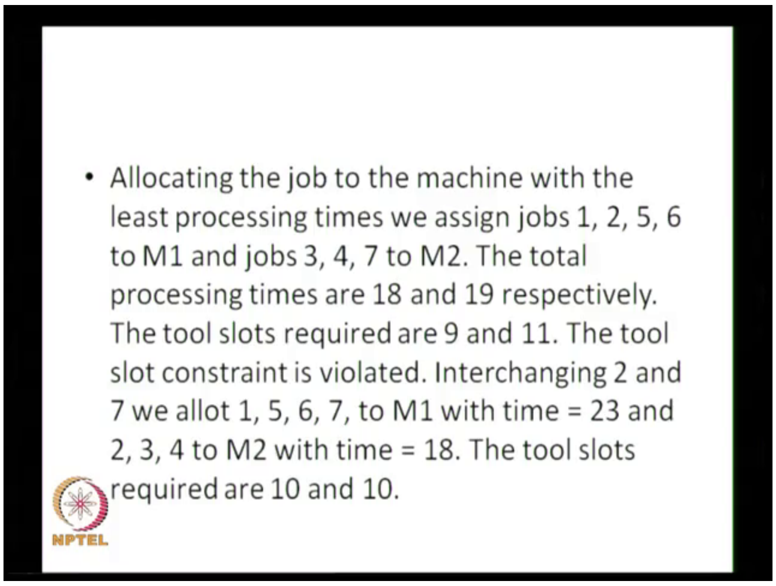
Now, we come here the load on M 1 is 1 2 3 and 5 1 plus 4 5 plus 7 12 plus 6 18 4 6 and 7 on M 2 is 7 plus 8 15 plus 7 22 22. So, again this has a solution with 22. So, it is an alternate optimum these all where has given 1 of these solutions with maximum value equal to 22, but on the other hand if we consider another feasible solution which is 1 2 3 and 6 if we consider 1 2 3 and 6 we do 4 5 and 7 . So, 1 2 3 and 6 again tools is satisfied. So, 1 2 3 and 6 is 4 plus 1 5 plus 7 12 plus 7 19 on M 1 4 5 7 on M 2 would be again 8 plus 7 15 plus 7 22. So, we again have another optimum solution. So, we have multiple alternate optima in this case the solver has given 1 4 5 7 on M 2 is 7 plus 7 14 plus 8 22.

So, we have this. So, it is given 1 of the alternate optimum which is this. So, the advantage of doing this balancing the workload to try and minimize the maximum is to ensure that in 22 time units in this particular solution all these 7 jobs are now completed and they can go to the next stage it is like minimizing a makespan, but all 22 jobs are completed. Now we could also look at in terms of if there are due dates then the problem becomes a little complicated because the sequence comes into the picture here we do not bring due dates into the account.

So, it is essentially an allocation problem this problem does not actually talk about at what time it 1 note of each of these 7 jobs are coming out it does not say that, it is only an allocation which says machine 1 is utilized for 22 time units machine 2 is utilized for 18 time units. And therefore, all of them since they are done in parallel all of them are over in 22 time units, but if we had individual due dates for each of these then we get into loading and scheduling problem which becomes little more complicated.

Now, which job will go to the other 1 is at and at what time what order will it go now the variable changes. So, it becomes a scheduling sequencing problem. Therefore, the variables now have to look at time particularly time, because we are talking of scheduling and units. So, when we bring due dates explicitly the decision variable also changes now we could think in terms of a simple heuristic to solve this problem.

(Refer Slide Time: 25:25)

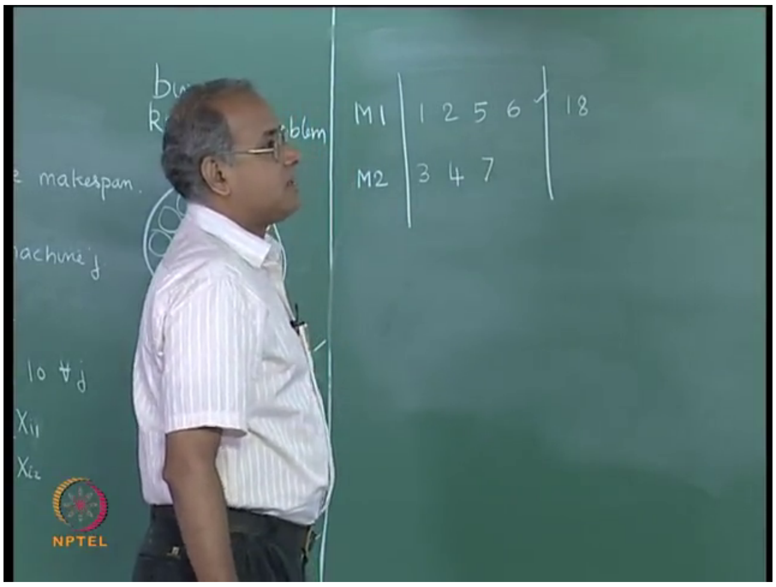


- Allocating the job to the machine with the least processing times we assign jobs 1, 2, 5, 6 to M1 and jobs 3, 4, 7 to M2. The total processing times are 18 and 19 respectively. The tool slots required are 9 and 11. The tool slot constraint is violated. Interchanging 2 and 7 we allot 1, 5, 6, 7, to M1 with time = 23 and 2, 3, 4 to M2 with time = 18. The tool slots required are 10 and 10.

So, a simple heuristic would be like this each job is now allocated to the machine where it has the least amount of processing time.

Now, we could look at 2 machines M 1 and M 2 a simple heuristic would be take each job and allocate it to the machine which has least processing time.

(Refer Slide Time: 26:15)



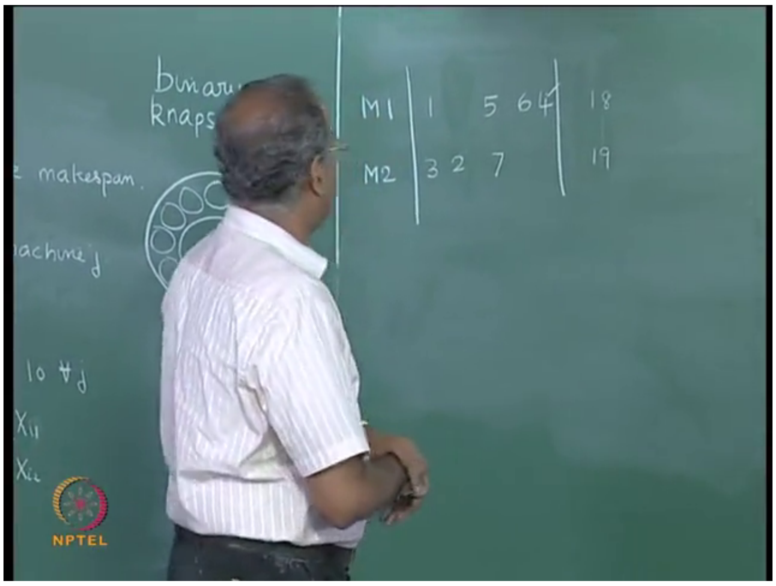
So, job 1 will go to M 1, job 2 will also go to M 1, job 3 will go to M 2, job 4 will go to M 2, job 5 will go to M 1, job 6 will go to M 2 job 6 will go to M 1 and job 7 will go to M 2 the 9 and 7. So, job 7 will go to M 2. Now we have to check whether this satisfies the tool requirement because if it does not satisfy the tool constraint then we would not able to run this whole thing in a single batch.

So, we look at 1 2 5 and 6 2 plus 3 5 plus 2 7 plus 2 9. So, this is satisfied 3 4 and 7 is 3 plus 4 7 plus 4 11. So, this is violet. So, we cannot implement this because for 3 4 and 7 put together the tool constraint is violated so, but let us just simply add the processing times to see where we are. So, 1 2 5 and 6 is 1 plus 4 5 plus 6 11 plus 7 18 and 3 4 and 7 5 plus 7 12 plus 7 19 if this were implemented or implementable then the load balance happens and at time equal to 19 all the 7 jobs are not, but unfortunately we are unable to implement this because the tool constraint is violated.

So, we could think in terms of either an exchange or we could think in terms of some other kinds of problems for example, we could now think in terms of taking something out of this. So, we had to remove 1 of these 3 or 4 or 7 such that we are we are within the 10 tool slots and we will look at 3 shifting it from M 2 to M 1 is going to increase the time by 2 units if we take M 4 it is going to increase by 1 unit and if you take M 7 it is going to increase by 2 units therefore, if at all we really want to take something out we can choose either the 1 which has minimum increase on the other 1 or in terms of tools.

So, let us try to put M 4 there. So, we now take this and go put M 4 here.

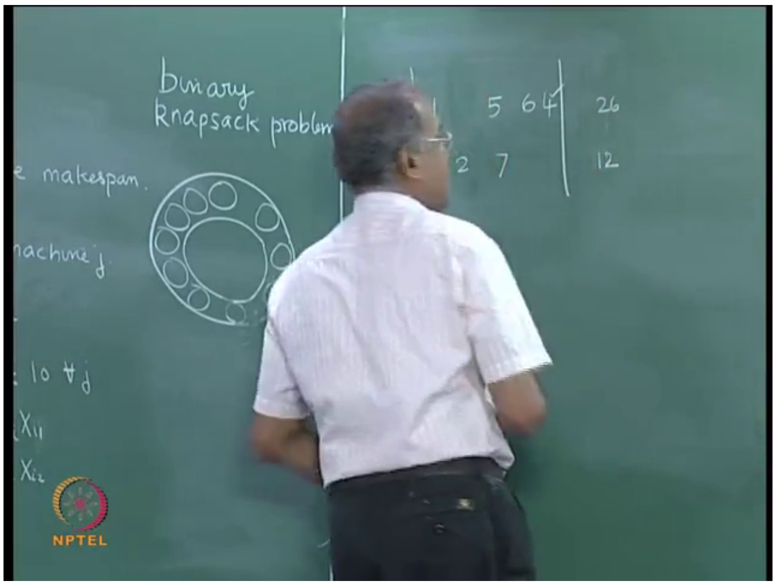
(Refer Slide Time: 29:10)



If we do that now 3 and 7 gives us time equal to 7, but 1 2 4 5 6 is 1 2 2 plus 3 5 plus 4 9 plus 2 11 plus 2 13 correct 13. So, from 1 2 4 5 6 we have to shift things here such that the time is minimum and whatever we shift we are able to transfer at least 3 a job that has 3 or more tool requirement has to go. So, we look at jobs that have 3 tool requirement job 2 has tool requirement of 3 fact job 2 is the only 1 that has tool requirement of 3. So, if we shifted then we shift by 2 units.

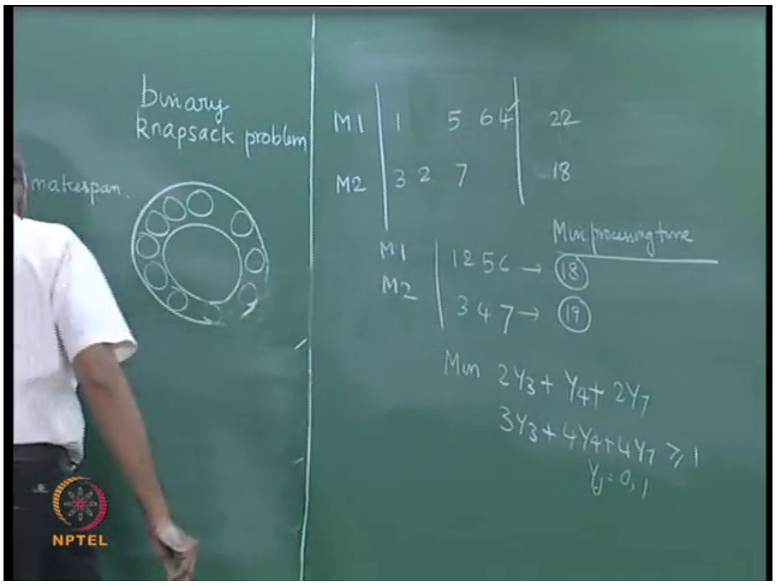
So, 2 3 7 will come here 1 4 5 6 will go to here if we do that the original values were 18 and 19 that were the original values. So, when we shifted it became the first shifted four. So, it became 18 plus 8 26 19 minus 17. So, it became 26 and 12.

(Refer Slide Time: 30:46)



Now, we shifted 2. So, 2 became here became 18 and here it became 20. So, we get 1 of the alternate solutions with 1 4 5 6 and 2 3 7. We did this for a very small problem we did this by looking at the problem, but we actually realized that if the problem is large and this itself has a large number of jobs.

(Refer Slide Time: 31:14)

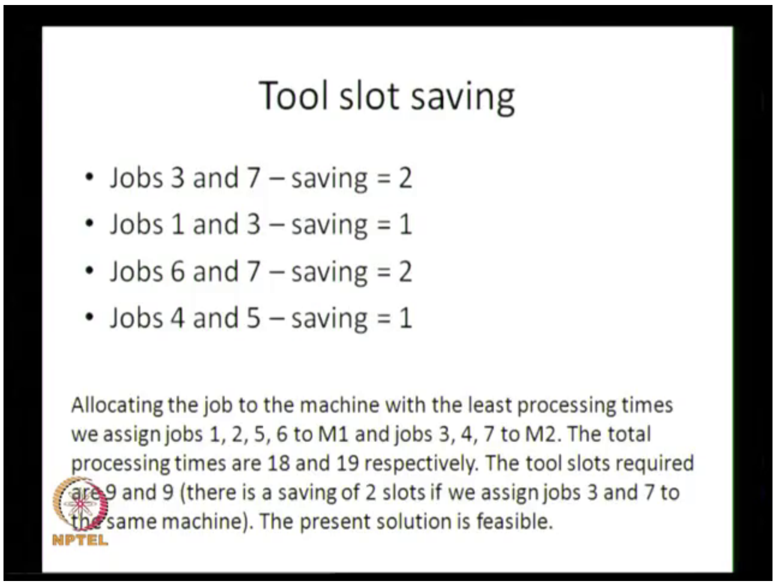


Then it is not only necessary the problem is a little tight in the sense that the some of the tools adds up to 20 the requirement adds up to 20 and the availability is also 20. Suppose the availability verse 22 and the requirements were 20 then this solution itself first solution itself would have been optimal or the point, I am making is when we actually move all that we need to do is we need not choose only 1 job at a time to move we could even choose 2 jobs at a time to move such that the total tools saved we are able to bring it back within feasibility.

So, that also becomes a binary knapsack problem which we actually solved in the part selection problem that is also similar to a binary knapsack problem. For example, if we choose to move if we when we started with this we started with 3 4 7 and 1 2 5 6 then when we did 3 4 7 we realized that the excess tool was 1. So, which ones to go here will be like this it will be like minimize if Y 3 equal to 1 if 3 leaves, Y 4 equal to 1 4 leaves, Y 7 equal to 1 if 7 leaves. So, if 3 leaves the minimum increase is 2. So, 2 Y 3 plus 1 Y 4 plus for 7 also it is 2 2 Y 7 such that job 3 requires 3 tools 3 Y 3, job for requires 4 tools 4 Y 4, job 7 requires 4 tools 4 Y 7 is greater than or equal to 1, because the excess was only 1 3 4 and 7 together have 11 tools, but then 10 is the availability. So, the excess is 1.

So, whatever goes out should take away at least 1 tool. So, this itself is a binary knapsack problem Y j equal to 0. So, we could solve a binary knapsack problem into it and then choose which 1 will go. So, here what I have done is I have given a solution where we interchange we could do interchange as a way to get it or we could solve binary knapsacks or we could simply allocate from 1 machine to another till we get feasibility. So, this is how the load balancing problem is solved in the context of F M SS, but we need to look at few more aspects of the load balancing problem.

(Refer Slide Time: 34:10)



Tool slot saving

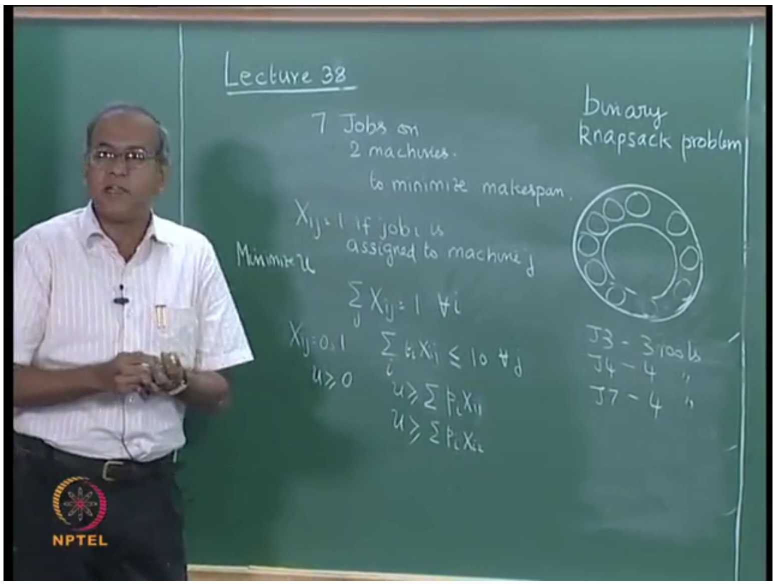
- Jobs 3 and 7 – saving = 2
- Jobs 1 and 3 – saving = 1
- Jobs 6 and 7 – saving = 2
- Jobs 4 and 5 – saving = 1

Allocating the job to the machine with the least processing times we assign jobs 1, 2, 5, 6 to M1 and jobs 3, 4, 7 to M2. The total processing times are 18 and 19 respectively. The tool slots required are 9 and 9 (there is a saving of 2 slots if we assign jobs 3 and 7 to the same machine). The present solution is feasible.

Now, we are introducing something that is that is very you need to F M S which is called tool slot saving, now let us go back to the same problem and if we had allocated the jobs based on minimum processing time minimum processing time, we would have got 1 2 5 6 for M 1 and 3 4 7 for M 2 we also found out that this had 18 total time this had 19 total time and the tool requirement was 9 and 11. Now we said we are not able to implement this because if I put jobs 3 4 and 7 together I need 11 tool slots from this we look at 3 4 and 7, I need 11 tool slots from this.

Now, let us go back and understand how it works here now job 3 requires 3 tools job 4 requires 4 tools and job 7 requires 4 tools.

(Refer Slide Time: 35:25)

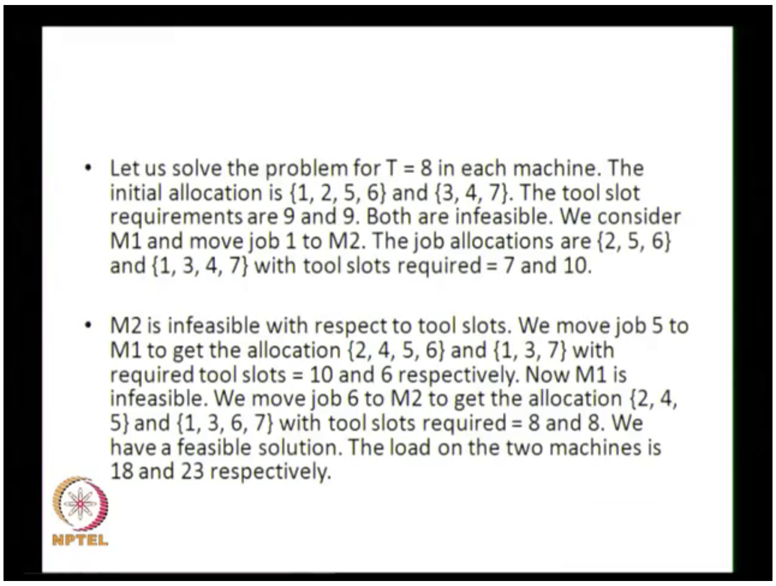


Now can we have a situation where right now when we say a total of 11 tool slots are required to put these 11 tools, we are going to assume that none of these tools are common? Now suppose we have a situation where these 3 tools and these 4 tools have 1 tool in common, suppose they have 1 tool in common then we actually do not need 11 tool slots we need only 10 tool slots. So, understanding that common tools and bringing them in can not only save the tools and tool slots, but it can give us a solution with 19 instead of a solution with 22 if we are able to find that amongst these 3 4 and 7 1 pair at least has 1 common tool then I do not have to I do not need 11 tool slots it is enough for me to do with 10 tools slots. So, that is what we want to show here when we do this with tools large savings.

Now, when we do this bit tool slots saving now we are going to assume that if we put 3 and 7 together we can save 2 tool slots which means 3 and 7 together they do not need 4 plus 3 7, but they need 5 4 plus 3 7 minus 2 1 and 3 put together I can save 1 tool slot 6 and 7 put together I can save 2 tool slots and 4 and 5 put together I can save 1 tool slot.

Now, when I start doing this now I know that the solution with 18 and 19 is feasible because based on minimum processing time I am allocated 1 2 5 6 to M 1 3 4 7 to M 2 3 4 7 together has 11 tool slots, but by putting 3 and 7 together I can save 2. So, I need only 9 tool slots. So, this solution is feasible, but what are the other issues that come when we do this.

(Refer Slide Time: 38:01)

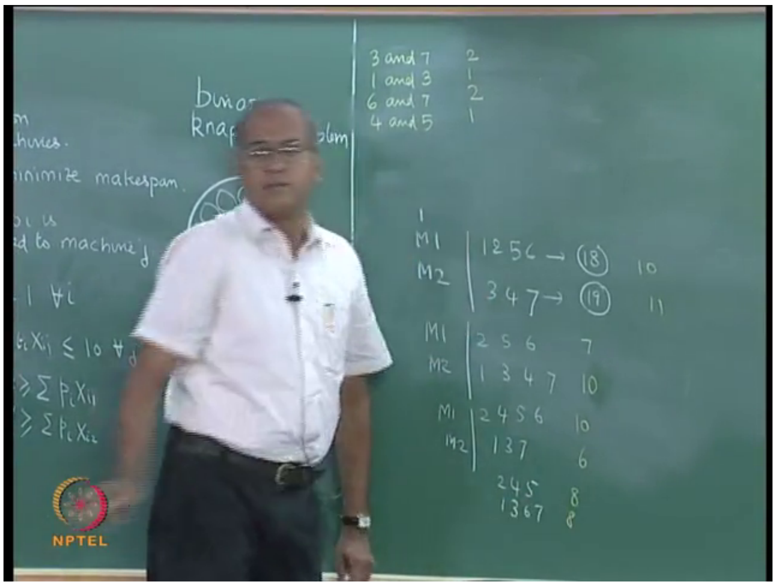


- Let us solve the problem for T = 8 in each machine. The initial allocation is {1, 2, 5, 6} and {3, 4, 7}. The tool slot requirements are 9 and 9. Both are infeasible. We consider M1 and move job 1 to M2. The job allocations are {2, 5, 6} and {1, 3, 4, 7} with tool slots required = 7 and 10.
- M2 is infeasible with respect to tool slots. We move job 5 to M1 to get the allocation {2, 4, 5, 6} and {1, 3, 7} with required tool slots = 10 and 6 respectively. Now M1 is infeasible. We move job 6 to M2 to get the allocation {2, 4, 5} and {1, 3, 6, 7} with tool slots required = 8 and 8. We have a feasible solution. The load on the two machines is 18 and 23 respectively.

Now, let us do let us continue and then let us look at the now let us look at a few other things to do this now let us instead of having 10 tool slots let us look at 8 tool slots, let us assume that this drum can accommodate only 8 tool slots instead of instead of 10 tool slots. So, we now begin with this is the minimum processing time solution this is the minimum processing time solution. Now this requires 10 tool slots this requires 11 tool slots to begin with let us also write these 3 and 7 I can save to 1 and 3 I can save 1 6 and 7 I can save 2 4 and 5 I can save 1 initial allocation is 1 2 5 6 and 3 4 7 that we have.

Now, we consider M 1 both are infeasible. So, we move job 1 to M 2. So, when we move job 1 to M 2 we get into another solution.

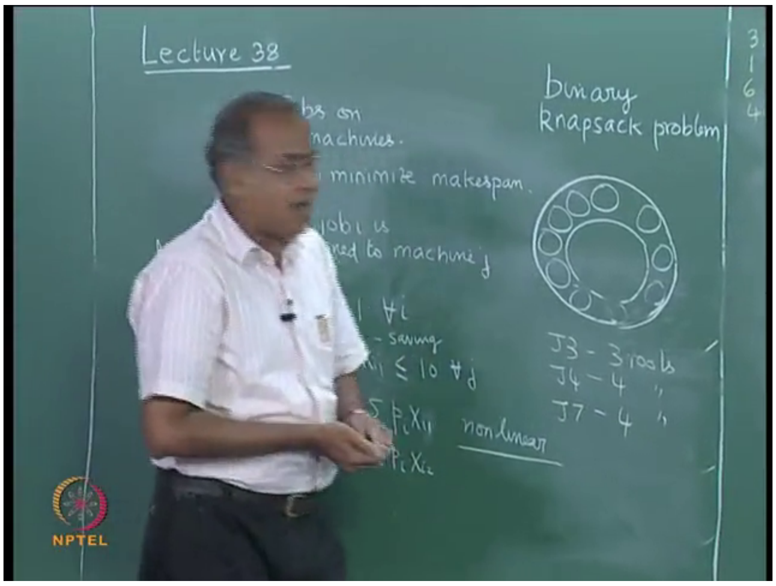
(Refer Slide Time: 39:50)



With M 1 M 2 2 5 6 1 3 4 7 2 5 6 2 5 and 6 7 7 slots 1 3 4 and 7 is 2 plus 3 5 plus 4 9 plus 4 13 are there, but 3 and 7 gives us 2 1 and 3 gives us one. So, this becomes 10 becomes 10. So, we get 7 and 10.

Now, M 2 is infeasible. So, we move job 5 to M 1 1 3 4 and 10 to get the allocation 2 4 5 6 and 1 3 7. So, we move job 4 to this. So, M 1 has 2 4 5 6 1 3 7 on M 2. So, we go back and check 2 4 5 6 2 4 5 6 is 4 plus 3 7 plus 2 9 plus 2 11 2 4 5 6. So, this is 10 1 3 7 is 3 again 2 plus 3 5 plus 4 9 9 minus 3 6 6, now you realize 2 4 5 6. So, 5 can be moved here to get 8 and 8. So finally, we get we get this solution 8 and 8. So, 2 4 5 1 3 6 7 6 can also be moved.

(Refer Slide Time: 42:02)



So, we can get 2 4 5 1 3 6 7 with 8 and 8 to get a solution with workloads 18 and 23 respectively. So, we could do this right.

Now, let us we look at the rest of them, but after we look at a few more things. So, let us come back here. So, given a certain tool drum capacity we can now have an algorithm which will finally, take us to that solution, but we will realize that this algorithm is not. So, very well defined in the sense that we simply looked at the problem and said we will shift 5 or we will shift 6 and so on, but each shifting we have to solve a single constraint problem.

Let us also look at a couple of related issues then we say that tools slot saving can now be incorporated. Now can we go back and solve this optimally now to solve this optimally this constraint will be t i X i j minus saving is less than or equal to 10, because the very advantage is the saving that we could consider, but then we realize that the saving cannot be very nicely represented here it will become non-linear I can save 2 if X 3 2 equal to 1 and X 3 7 equal to 1.

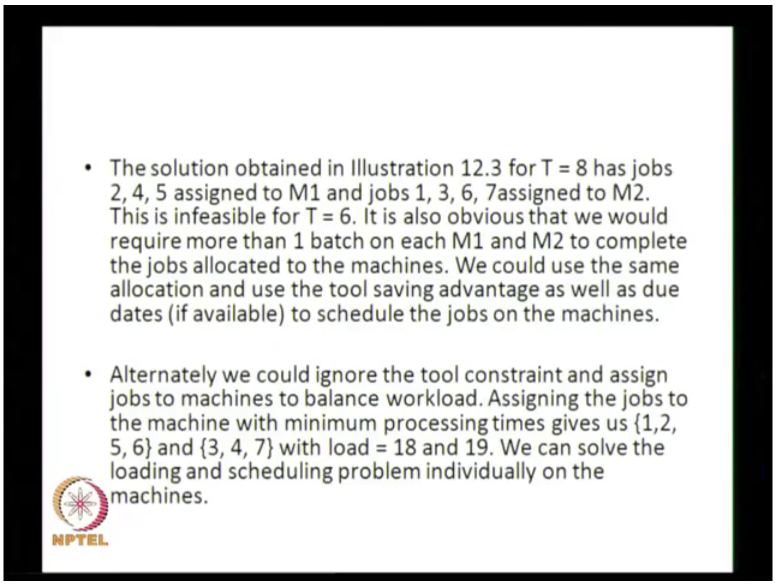
So, the saving will be 2 into X 3 2 into X 3 7 only when both of them are 1 I get the saving if 1 of them is 0 I do not get the same. So, it becomes non-linear. So, the problem becomes non-linear that is the first part. Similarly if I am looking at moving 1 from this to the other or from the other 1 to this and if I end up looking at the problem once again as a knapsack problem, once again I get a non-linearity coming in because then I move 1 then next 1 it again has a non-linear term because I gain by when I move 1 jobs from M 2 to M 1.

Now, I save something on M 2 at the same time I have to go to M 1 and also look if there is a saving because a new job comes in and not only that when I actually release something from M 2. For example, if I am releasing job 6 from M 2 by releasing job 6 if job 6 alone requires if jobs for example, if I have a situation where, let us say I am releasing job 7 from M 2, I am releasing job 7 from M 2 to M 1 by releasing job 7 from M 2, I am not exactly saving 4 slots because if 3 and 7 where in M 2 together I have already saved 2 slots by putting 3 and 7 together.

Now, by releasing 7 3 would want those common tools. So, I am not saving 4, but I am saving only 2 slots because 2 out of the common tools are still required. So, that also has to be brought in when we actually move jobs from 1 to another. So, nonlinearity will create a lot of issues when we do this particularly either when we try and solve this here or we try and solve this here it is not impossible to do that, but nonlinearity will bring in issues.

Now, the knapsack problem has to be modified. So, there is a branch and bound algorithm which has been published by Barada and stick which takes into account this nonlinearity and then it develops a branch and bound algorithm to do load balancing considering tools slots saving also, but 1 has to understand that the nonlinearity will come in here as well as here, because when we release we do not actually save for slots, we save less than that if 3 and 7 had belonged to job 2. Similarly when 7 goes somewhere we then need to adjust these not requirement there which is an important aspect.

(Refer Slide Time: 46:55)



- The solution obtained in Illustration 12.3 for T = 8 has jobs 2, 4, 5 assigned to M1 and jobs 1, 3, 6, 7assigned to M2. This is infeasible for T = 6. It is also obvious that we would require more than 1 batch on each M1 and M2 to complete the jobs allocated to the machines. We could use the same allocation and use the tool saving advantage as well as due dates (if available) to schedule the jobs on the machines.
- Alternately we could ignore the tool constraint and assign jobs to machines to balance workload. Assigning the jobs to the machine with minimum processing times gives us {1,2, 5, 6} and {3, 4, 7} with load = 18 and 19. We can solve the loading and scheduling problem individually on the machines.

Now, continuing on this further now there are there is also in in practice there is also a due date dimension these jobs will have to go to the customers at certain specified due dates. So, the moment we bring in due dates we get into a typical scheduling problem where time of completion becomes a variable and the problem becomes complicated. Secondly, we also if we look at this particular problem that we have example that we have solved we have 7 jobs and the way this problem is created a total of 20 tool slots are let us even forget that we are going to look at tool saving constraint.

Let us look at the problem as it is without a tool saving constraint no total tool slots required is 20, total tool slots available is 20. If there were tool saving they could even be variable to solve when we had 8 tool slots available which means with 16 availability and 20 requirement and 4 saving we were able to get this, but suppose we have about 20 jobs and we have 2 machines.

Let us say the tool capacity is 10, but these 20 jobs put together require about 40 tools number 1 is we may not actually have we might have only 20 tools corresponding to the tool drum, the point is now these jobs have to be allocated to these machines and then let us assume for the sake of discussion that the first 10 jobs go to M 1 and the next 10 jobs go to M 2 the requirement of tools for the first 10 jobs would exceed 10.

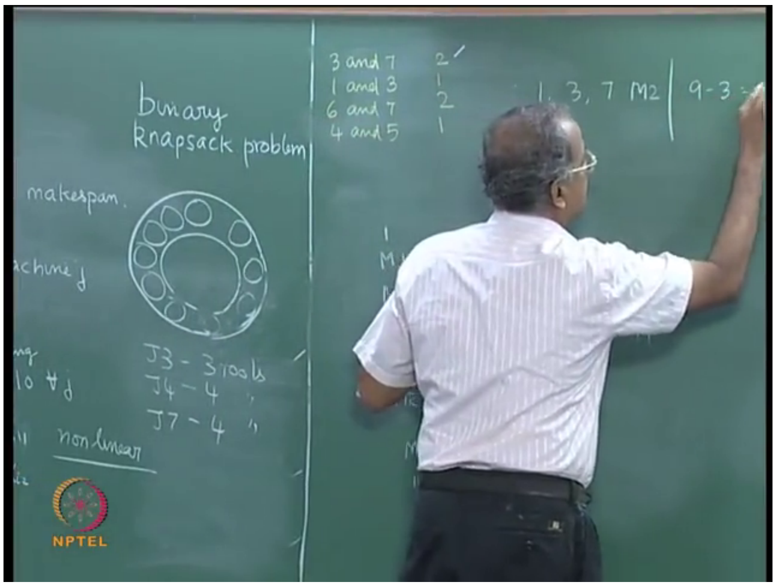
So, the first 10 jobs will have to be run on multiple batches because the 10 tool slots will now be set up a subset of these 10 will be done with this and then the second part of that

10 will have to be done again when these tools slots are either retained or some changes are made to this depending on the requirement of the other jobs.

So, there is also a batching problem. So, when multiple jobs are there on machines, which job is allocated to which machine, which job will go to which batch, called the reason is between every batch some tools will be removed and some tools will be brought. So, there is a tool change over time between every batch. So, which batch will it go and then will it be scheduled and if there are due dates there is a due date problem there is a batch batching problem and there is a load balancing. So, there are 3 different problems that come in and in addition to that there is a nonlinearity that comes in when it is possible to have changes tools slot when it is possible to gain 2 tool slots if jobs have common tool requirements.

Now, this common tool requirement is also something interesting now we could say that 3 and 7 gives us saving of 2 1 and 3 could give us a saving of 1 in our calculation.

(Refer Slide Time: 50:13)



If we put 1 3 and 7 to the same machine we will go back and say 1 3 and 7 requires 9 tool slots there is a saving of 2 here there is a saving of 1 here. So, 9 minus 3 is equal to 6 it could be 6 it could sometimes be 7 because it might happen that 3 and 7, I can save 2 slots 1 and 3 if it requires the same tool slot then can I subtract or if 1 3. Now it depends on which are the tool numbers which are the tool numbers is another question that we could have to look at sometimes we may actually have to define them exactly in terms of tool numbers and then see what is the saving. So, when we actually do this when 2 of them have savings there could be an intersection that comes which has to be added

So, it we the 1 may not be an effective tool slot saving. So, that can also happen. So, the nonlinearity is actually further compounded by multiple. For example, when I do 1 3 and 7 I might say I will save. So, much if I do 1 and 3 I have this much saving I have 3 and 7 I have this much saving when I do 1 3 and 7 on the same machine it may not be the sum of these 2, but it could be something else. So, the nonlinearity is further compounded. So, a machine loading problem becomes an interesting problem and more involved when we have multiple batches, when we have tool slot saving, when we have due date related measures. So, this problem is necessary and useful. So, that the time taken to produce on an FMS is reduced by properly solving the machine loading problem.

Other problems related to flexible manufacturing, we will see in the next couple of lectures.