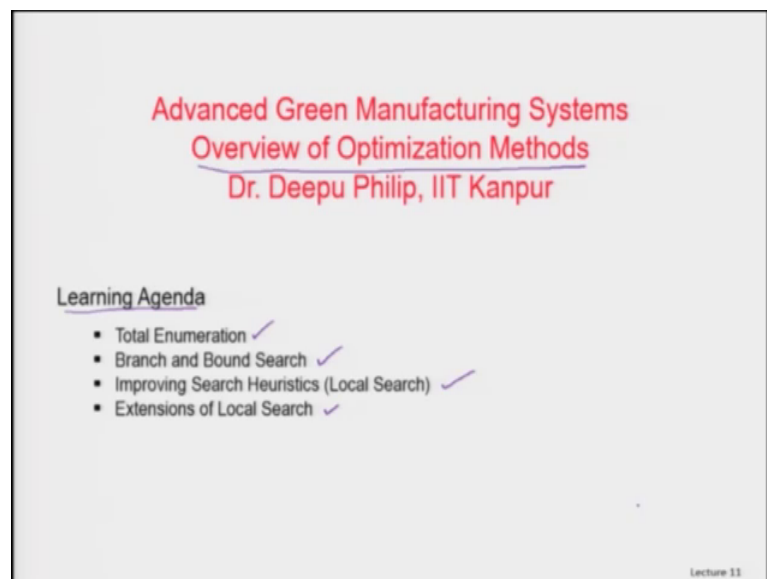


Advanced Green Manufacturing Systems
Prof. Deepu Philip
Dr. Amandeep Singh Oberoi
Department of Industrial & Management Engineering
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

Lecture - 39
Overview of Optimization Methods – Part 1

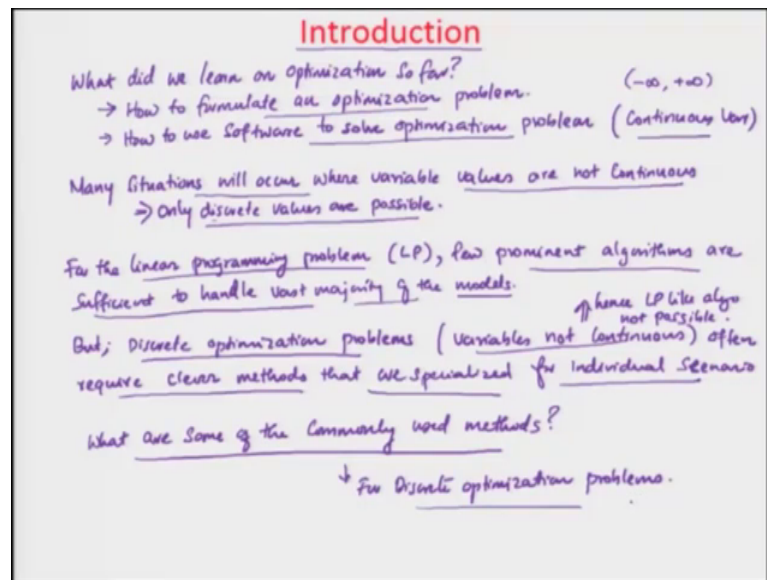
Good afternoon students. Welcome to one other lecture in the Advanced Green Manufacturing Systems. We are coming to the close of the course most of we have covered most of the topics and with this lecture we would probably now cover one of the major concepts that we have trying to do is the optimization and some of the algorithms associated with it. So, without wasting too much of time I will just jump into today's topic.

(Refer Slide Time: 00:42)



So, if you look at today's presentation we have overview of optimization methods is what we are going to do. We seen how the optimization problems are and now we are trying to look into what are some of the methods how do we solve this. And today's learning agenda we look into something called total enumeration and branch and bound search, then improving search heuristics which is called local search and the extensions of local search. These are the four concepts that we will cover today.

(Refer Slide Time: 01:05)



So, first let us talk about this new concept that we are going to cover today the class. So, what have we learned so far? What did we learn on optimization so far, what did we do? The thing that we learned so far is how to formulate an optimization problem? How to formulate an optimization problem this we have seen.

We are good at identifying how are we formulating this; this we know how do we do that and what are the other aspects of it and we have a good idea. Then we also saw how to use software to solve optimization problem ok.

Here in this case we have been limited to what you called as continuous variables ok. So, our variables are continuous in nature continuous variables; that means, they are vary from X vary from minus infinity to plus infinity ok. This is what our concept was and we know how to formulate those problems. And we learn how to solve them using software packages right.

The trick is that many situations, many situations will happen where many situations will occur, where variable values of the variables variable values are not continuous ok. These variables are not continuous or in other way only discrete values are possible. So, what we are saying is that many situations, many of the optimization problems will not have the variables has continuous variables. Instead only discrete values are possible discrete means integer values countable values those kind of things.

So, for the linear programming problem or what we call as LP ok where the variables are continuous and step like that ok. Few prominent algorithms few prominent algorithms are sufficient are sufficient to handle vast majority of the models of the optimization models or optimization problem.

So, for LP problems where the variables are continuous there are few algorithms available they are prominent algorithms and they are sufficient, they are very good in handling majority of these models ok. But the major important thing for this is, but discrete optimization problems discrete optimization problems. What are discrete optimization problems? Variables are not continuous variables, not continuous your variables are not taking a continuous value ok.

This discrete optimization problems often require clever methods that are specialized are specialized for individual scenario ok. So, for each individual scenario specialize specific scenario of the discrete optimization problem, we require clever methods that are specialized. Highly specialized clever methods are require for discrete optimization problem the reason being is variables are not continuous. So, hence LP like algorithms not possible not possible; we cannot use those algorithms to handle such kind of problems ok.

So, what are some of the commonly used methods or question today is this is what we are going to learn today. What are some of the commonly used methods on which context for discrete optimization problems and this is very common because in real life in manufacturing you do discrete manufacturing most of the time. You make cars, you make a parts, you make engines all those kind of things they are all discrete optimization problems because they are countably discrete there is no continuous values there.

(Refer Slide Time: 06:40)

Total Enumeration

It is often counter intuitive that discrete optimization problems are more difficult than their continuous analogs.

Because it appears that only a finite number of choices are possible for decision variables; hence should be easy.

Hypothesis-1: If a model (Discrete opt) has only a few discrete decision variables, the most effective method of analysis is enumerating all possibilities to obtain optimal solution.

This approach is known as total enumeration.

— Checking of all possibilities implied by discrete variable values are conducted, then choosing the best combination that gives the best objective function value.

very few ↑

So, the first and for most algorithm that we are going to talk about is total enumeration. And you would not believe this is a pretty effective thing for small cased problems. So, like one of the things is it is often counter intuitive, you can it is often counter intuitive that discrete optimization problems. What are discrete optimization problems? Problems where the decision variables are not continuous, problems are more difficult are more difficult than their continuous analogs.

So, what we are saying is that the discrete optimization problem for a particular case is more difficult to solve than the same continuous their analog of the continuous the analog problem which is of where the variables are in continuous. Why because it appears that it appears that only a finite number of choices are possible are possible for decision variables for decision variables. A finite number of choices are possible for decision variables; hence should be easy.

So, you think that in a continuous problem the variable can take infinite number of choices any value it is possible. Where as in a discrete problem there are only limited values a finite number of values are possible for the decision variables hence you should be ideally easy, but this actually counter intuitive because it is not easy. Because in when the variables are continuous the most easiest thing is that you know you can use things like mathematical things like differentiation, integration or slop those kind of things

gradient like techniques. Where as in these kind of problems it there kind of a thing is not possible ok.

So, that is a one hypothesis one. Let us think about a important hypothesis in this regard if a model that is a discrete optimization model has only a few discrete decision variables; discrete decision variables. If a model has only few discrete decision variables the most effective way or effective method of analysis is enumerate all possibilities to obtain optimal solution ok.

So, what we are saying here is if the problem the model is a discrete optimization model and there are only few decision variables ok. Number of decision variables are very small, then the effective way of getting the optimal solution, how can you find the optimal solution effectively is to enumerate all possibilities went through all possibility options and then find out ok. This method this approach is known as total enumeration.

So, when somebody says total enumeration then you should understand few things one is the decision variables have only few discrete few discrete options. And then out of this you are able to get an optimal solution.

So, this total enumeration where we define it as checking of all possibilities you are checking of all possibilities implied by discrete variable values or taken by discrete variable values are conducted. You do all discrete variable values you take them and you check all those possible values and then choosing the best combination that gives the best objective value that gives the best objective function value.

So, what you are trying to do here is you check all possibilities implied by the discrete variable whatever the decision variables remember these are very few in number ok. These few discrete decision variables, you find out what are the all possibilities values it can take, then what you do is you choose the best combination of this that gives the best objective function value. Whatever is the best combination you take that in you call it as the best objective function value that is the idea fine.

(Refer Slide Time: 12:43)

Example of Total Enumeration

Maximize: $7X + 4Y + 19Z$ (objective function)

Subject to: $X + Z \leq 1 \rightarrow ①$
 $Y + Z \leq 1 \rightarrow ②$
 $X, Y, Z \in \{0, 1\}$

Constraints: The decision variables (X, Y, Z) can have either '0' or '1' as its values.

Three variables; each having two values possible $\Rightarrow 2^3 = 8$ possible choices.

Possible choices: (X, Y, Z)	Constraints: $(X, Y, Z) \in \{0, 1\} \rightarrow$ satisfied all cons.
$(0, 0, 0)$	① ✓ ② ✓ \rightarrow check which all satisfy?
$(0, 0, 1)$	① ✓ ② ✓ \rightarrow check which all satisfy?
$(0, 1, 0)$	① ✓ ② ✓ \rightarrow check which all satisfy?
$(1, 0, 0)$	① ✓ ② ✓ \rightarrow check which all satisfy?
$(0, 1, 1)$	① ✓ ② ✗ \rightarrow infeasible on ②
$(1, 0, 1)$	① ✗ ② ✗ \rightarrow infeasible on ①
$(1, 1, 0)$	① ✓ ② ✓
$(1, 1, 1)$	① ✗ ② ✗ \rightarrow infeasible on ①

So, let us see a simple example just look at a example of a total enumeration ok. So, this is kind of like a tricky problem, but I think it will make. So, let us write a linear programming problem maximize that is a objective function $7 X$ plus $4 Y$ plus $19 Z$ ok. And subject to the constraint is X plus Z is less than or equal to 1 ; then Y plus Z is less than or equal to 1 ; then X, Y and Z they are element of $0, 1$ ok.

So, what you are saying is that you are three variables ok. These are the coefficients of the variables this is the objective function ok. Objective function and these are your constraints right. So, then; obviously, the thing is that the most important thing that you should understand is X and X, Y, Z the decision variables ok. The variables on which you can take decisions X, Y and Z can half either 0 or 1 as it is values. So, you can have only either 0 or 1 in this particular case for the decision variables X, Y and Z ok.

So, since there are three variables how many possibilities are there? Three variables each having two values possible which gives you 2 to the power of 3 8 options; 8 possible choices. So, 8 possible choices this is manageable by you. So, what are the possible choices let say the possible choices ok.

The possible choices in our case are let us call it as of X comma Y comma Z is a possible choice is a $0, 0, 0$ that is one possible choice. So, that is means all the values are 0 then the other possible choice is $0, 0, 1$ ok. So, like Z taking value of one-third choice is $0, 1,$

0; where Y is taking this 1. Then you can have is 1 0 0 that is another possible combination, so each one of them taking this particular value.

Then we can have another value as the 0 1 and 1, Where both Y and Z are taking the values of 1 is 2, 4 and 5. So, then the other value that is possible is 1, 0, 1 X and Z taking 1 and Y taking 0. Then the other value there is possible is 1 1 and 0 right and so now you have 2, 4, 6, 7 and the last possible value is all taking 1 right. Now let us see in these cases what are the values of the objective function right. So, now let us say constraints constraints ok.

So, let us see if you put the value of 0 and 0 ok. So, let us see the first constraint the two constraints right. So, this satisfies the X Y and Z these variables element of 0, 1; this is satisfied in all cases ok. So, then the other constraint which let us call this as constraint 1 and let us call this as constraint 2 right, so the X plus Z less than or equal to 1 is it satisfied in all cases. So, this is X, Y and Z. So, this is X plus Z less than or equal to 0, so 1, 2 ok, so 1 is satisfied.

So, we have put one two there is also first constraint second constraint; first constraint second constraint; first constraint second constraint; first second first second. So, there are two cases in this regard so the case of constraint one check which all satisfy.

So, we check the first constraint which one of them X plus Z less than or equal to 1 is that satisfying. So, 0 plus 0 less than or equal to 1 that is correct this case is X plus Z. So, 0 plus 1 less than or equal to 1 this is also satisfied; this is 0 plus 0 less than or equal to 1 satisfied; 0 plus 0 less than or equal to 1 satisfied. This is 0 plus 1 less than or equal to 1 satisfied; again this is 1 plus 1 that is 2 less than or equal to 1 not satisfied; 1 plus 0 less than or equal to 1 satisfied 1 plus 1 less than or equal to 1 not satisfied.

So, these two cases immediately you can say that fine I do not need to deal with this because this is infeasible on 1; the first constraint is infeasible here is also infeasible on 1. So, with the first constraint we know that these two are not needed.

So, then we do not need to check the second constraint at all because you this problem you can eliminate. So, now immediately from 8, you are now reduced to 6 of them. Now let us look at the second constraint which says X sorry Y plus Z less than or equal to 1 this is the check which all satisfied.

So, now we are checking which all will satisfy Y plus Z less than or equal to 1. So, Y plus Z less than or equal to 1; 0 less than or equal to 1 this satisfies Y plus Z . So, 0 plus 1 is 1 ; 1 less than or equal to 1 satisfies 1 plus 0 less than or equal to 1 satisfies 0 plus 0 less than or equal to 1 satisfies right and then 1 plus 1 this is less than 2 less than or equal to 1 does not satisfy right. Then in this case 1 plus 0 less than or equal to 1 satisfies.

So, we get this particular case also infeasible on 2. So, of these cases we know that we do not need to worry about this case, we do not need to worry about this particular case and we do not need to worry about this particular case. So, out of 8; 1, 2, 3, 4, 5 these are the 5 possible cases we just need to now check.

(Refer Slide Time: 20:10)

Enumeration Example ...

Evaluation Case:	Objective Function Value
$(0, 0, 0)$	0
$(0, 0, 1)$	19 ✓
$(0, 1, 0)$	4
$(1, 0, 0)$	7
$(1, 1, 0)$	11

Since the objective was to Maximize
Choose the largest
Value of Objective
Function.

Calculations

$(0, 0, 0) \Rightarrow 7 \times 0 + 4 \times 0 + 19 \times 0 = 0$
 $(0, 0, 1) \Rightarrow 7 \times 0 + 4 \times 0 + 19 \times 1 = 19$
 $(0, 1, 0) \Rightarrow 7 \times 0 + 4 \times 1 + 19 \times 0 = 4$
 $(1, 0, 0) \Rightarrow 7 \times 1 + 4 \times 0 + 19 \times 0 = 7$
 $(1, 1, 0) \Rightarrow 7 \times 1 + 4 \times 1 + 19 \times 0 = 11$

Evaluated limited and ^{all} feasible
alternatives of decision variables.

Value of $X = 0$
Value of $Y = 0$
Value of $Z = 1$
Objective Function = 19

Optimal Solution for
the problem:

So, our possible cases in this include ok. So, now if you continue with this working of this example the possible cases are now the evaluation cases. The cases that we need to evaluate in this case include $0, 0, 0$ ok. Then we have is $0, 0, 1$ then we have is $0, 1, 0$; then the next one we have is $1, 0, 0$ and we have is $1, 1, 0$. So, these are the 5 cases if you look into this we got these values from this particular table.

So, we are only evaluating these cases ok. So, now, how do we evaluate that? So now we do is these we just need to find the objective function value ok. Because these are feasible, these conditions of the mar feasible you have already check the feasible. Now we just need to find what the objective function value is. So, how do we calculate the

calculations include like this. So, for 0, 0, 0 it is equal to you know 7 times 0 plus 4 times 0 plus 19 times 0 which will give you 0.

So, the objective function value is 0 in this case ok. The other case is let us take 0, 0, 1 then it will be 7 times 0 plus 4 times 0 plus 19 times 1 that will be 19. So, 0, 0, 1 will give you the value of 19 right, then you have 0, 1, 0; 0, 1, 0 will be 7 times 0 plus 4 times 1 plus 19 times 0 will give you 4 ok. So, 0, 1, 0 the objective function value is 4.

Now 1, 0, 0 will give you 7 times 1 plus 4 times 0 plus 19 times 0 will give you 7, so this value is 7 right. Now the last one is 1, 1, 0 so that will be 7 times 1 plus 4 times 1 plus 19 times 0 is 7 plus 4; 11, so our objective function value is 11. So, these are the four possible objective function values.

Now, the objective was to maximize ok. Since the objective was to maximize choose the largest value of objective function ok. So, what we have doing is from here this is the largest value of objective function. So, we say this is the best possible solution for this.

So, the solution is that so what we are saying that the value of X is equal to 0; value of Y is equal to 0; value of Z is equal to 1 ok. Objective function is 19 and this is the best possible set of for, this is the optimal solution optimal solution for the problem ok. So, depending on the constraints we already seen that we eliminated infeasible solutions these are all infeasible.

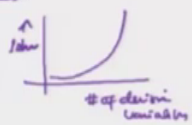
So, we eliminated them you using both constraints and then only the remaining ones were work out to find out what the best possible solution is; this where we so we evaluated limited and yet feasible alternatives of decision variables ok. So, we evaluated and yet feasible and yet all feasible alternatives of decision variables were done and hence this why it is called as total enumeration ok.

(Refer Slide Time: 24:41)

Branch and Bound Search

What is the major drawback of total enumeration?

- not feasible for a large set of decision variables.
- The exponential growth of solutions:
if 3 decision variables $(0, 1) = 2^3 = 8$
if 4 " " $(0, 1) = 2^4 = 16$



Alternative? Hypothesis - 2

if we have a way to deal with problems having large set of decision variables \rightarrow can the optimal solution be found without explicit enumeration of all possible & feasible cases.

Branch & Bound: - Subset enumeration strategy - Systematically form classes of solutions and investigate whether the classes can contain optimal solution.

\Rightarrow usually the tree search approach is used to expand optima.

Hopefully this makes sense to you; from there we get into the next algorithm or next approach what we called as branch and bound search ok. This is also a very popular approach among lot of people now let see what it is ok. What is the major what is the major drawback of total enumeration? Major drawback of total enumeration, what happens when you do the total enumeration? Why is the; what is the major drawback why is it like people do not heavily use it ok.

This is not feasible for a large set of decision variables. So, if you have a large set of decision variables this is not very feasible. Why? The exponential growth of solution space of solutions ok so for example, if 3 decision variables let say 0, 1; then it is 2 to the power of 3; 8.

Let us say if four decision variables again 0, 1 then it is 2 to the power of 4. What is that 2 to the power of 4 will be 16 ok? So, now if it becomes 5 then it will be 16 times 2; 32, so it will grow. So, this solutions pays will grow something like this, this exponential growth if this is the number of decision where number of decision variables ok, this is the solutions.

If you draw graph like this then the growth will be like an exponential growth. So, then when it is large number of decision variables. Now you have to look at if it there a 5 then you will look at 32 options pretty soon it becomes manually impossible for you to evaluate the things ok.

So, then alternative what else can we do ok. What is alternative to this headache ok? So, this is the let us say hypothesis 2, second belief system that. If we have a way to deal with problems having large set of decision variables set of decision variables ok. We have problem, which has large set of decision variables can the optimal solution be found without enumerating without explicit enumeration of all possible excuse me and feasible cases.

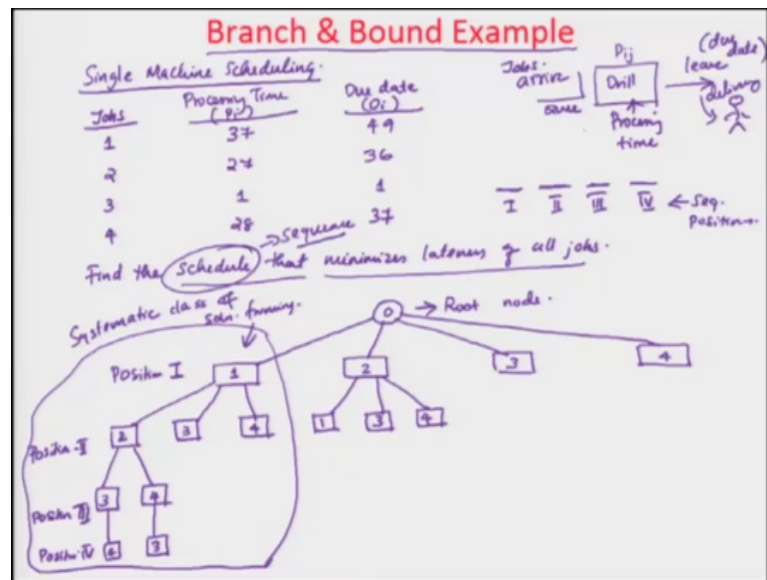
So, what we are trying to say or our belief is that if there is a way, if there is a method available where problems having large set of decision variables are there they are all discrete decision variables ok. Where we still want to find the optimal solution we want to get the optimal solution that is provably optimal while we do not want to enumerate all possible and feasible cases ok.

So, then branch and bound is one such strategy so, we can we call it as with is a sub set enumeration strategy ok. So, we are saying is that we do not want to enumerate every possible options ok. So, what we say is that systematically form classes of solution systematically from class solutions and investigate whether the classes can contain optimal solutions.

So, what you are saying is from the total problem you create systematically from the classes of solution and see whether these classes can contain optimal solution. If we find that the classes cannot contain optimal solutions then do not look into that class. We find that the class can come have the optimal solution then look into this so that kind of an approach.

So, usually the tree search approach is used to expand options so or classes ok. So, what we do is we use a tree search approach in this regard to see whether we can actually systematically form the classes of solution and then investigate whether the class can contain the feasible solution or not ok.

(Refer Slide Time: 30:41)



So, again as I said earlier let us look at a simple example of this one this would make a you will understand the problem better ok. So, this is a single machine scheduling. So, the idea is that you have a machine a single machine let us call it as drill and you know entities arrive. So, they arrive into the system this is the que or something. And then they get drilled here then there is a processing time P_{ij} ok, whatever the processing type and then they leave and this is the machine ok.

So, what happens is there are like say and jobs arrive there are different jobs are available. So, let us call the number of jobs for this case as 1, 2, 3 and 4. There are four jobs available the processing time p_i . For 1 let us make the processing time as 37, for 2 the processing time is 27, 3 the processing time is 1 and the 4 the processing time is 28 ok.

And then there is one more that is called as due date where you are supposed to deliver the so this is the processing time, this is associated with this. Then here is a customer sitting here and here is the delivery and this is where the due date happens due date means when you have to give it to the customer. So, in this case let us put a value of 49, let us put 36, let us put 1 and 37, now the question here is ok. So, we can schedule it in many ways, but find the schedule that minimizes lateness of all jobs ok.

So, what you are trying to do is we want to find the production schedule that minimizes or find the sequence or schedule. Let us call it as a sequence for the time being schedule

and sequence there is only much small difference. So, sequence means there are four jobs. So, there is roman numerals are sequence positions ok. These are sequence positions and we have to decide which job we will go into which sequence position in this regard right. So, you can solve it as a discrete optimization problem as well or you can use branch and bound to solve a case.

So, one way to think about this problem is that ideally speaking you can think about it as a tree search. So, if you say you call this as 0, let us call this as root node for the time being and let us call this as the position 1 ok, position 1. So, then you can have four possible options out of this can put four possible jobs ok. So, these are the four possible jobs.

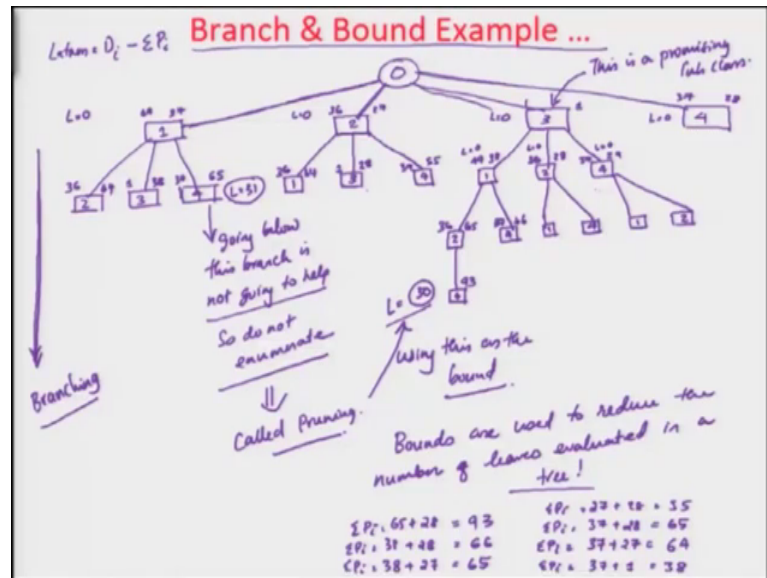
So, if you put 1 here, 2 here, 3 here and 4 here. So, that is equivalent to saying sequence position 1 is taken by 1, 2, 3 and 4 ok. Then we take from here once you fix this when you say I am going to gum to this particular portion then you can say that there is another position called position 2, sequence position 2 that position 2 in this job we can have 3 under need them ok.

So, once one is fixed then you can have for the position it is 2, 3 and 4 any one of them can take the sequence position 2, for 2 again there is possible possibility of 3 ok. So, this is 1, 3 and 4 you cannot use 2 because 2 is already gone in to the first sequence position right.

So, then if you think about this below that you have something called as position 3, the sequence position 3. There are only 2 possible options for this for this case what because you have already used 1, you have already used 2, then the possibilities are 3 and 4.

So, then the last case is position 4, this can have only 1, it can be 4, this can be 3. So, when you think about this the sequence becomes 1, 2, 3, 4 and the other sequence is 1, 2, 3, 4. So, what you have done is; this is where you have created something called as a, what I mentioned earlier this particular term is systematically form classes of solution ok, systematic class of solution forming that is what we are doing here in this particular case. So, how would this actually help us in solving the problem?

(Refer Slide Time: 36:21)



So, let us see an example ok, this same example let us workout little bit longer. So, that would probably help us in a understanding the mechanics of the problem. So, our root node is 0, and we have our four sequence positions ok. I hope we get space here 1, 2, 3 and 4 ok.

So, the first sequence position is taken care of in this case we have made the first sequence position set up to 1, 2, 3 and 4. Now, you can decide the minute you put one here the time taken to produce the p_i for. So, the lateness late let us call lateness equal to due date D_i the job minus sum of P_i 's all the processing times.

So, for this case the lateness is equal to what is the value of lateness, the due date the processing times are 37, 27, I am just writing it here for the time being making my life easy 1 and 28 ok. So, for 37 if you think about it the due date of this problem is I am writing the due date on this side which is 49, this is 36, then this is 1 and 37 ok. So, in this case the lateness is 0, the in this case also the lateness is also 0 ok. I am just putting 1 equal to 0 here. This is also lateness equal to 0 lateness equal to 0 fine.

Now, there are multiple ways you can think about ok. So, sometimes what people do is fine I will take one of these sub class and expand it and so that I get a solution out of it and see wherever I exceed this I do not continue this problem. So, let us look at that example so I am just going to take this branch one and expand out of it right.

So, the next one I am going to do is the sequence position 1, so you finished to that. So, now, let us think about the next job that we are going to put in here we have 3 jobs here possible there is 2, 3 and 4 for us. So, now the processing time of job 1 is 37, if we put 2 here ok. So, then that becomes the type two processing time is which is two processing time is 27.

So, the time taken is the P_i , σP_i in this regard is here will be I will kind of use the bottom portion the σP_i is equal to 37 plus 27 right. So, that will give you 7 plus 7 is 14 3 plus 2; 5 plus 1 6. So, this will be 64; 64 and 2 is supposed to be delivered on 36th day and you are already late now ok.

Now, about 3 if you think about it 3 the time is 1. So, the σP_i for 3 will be 37 plus 1 that is 38 right. So, this will be 38 and 3 is supposed to be delivered on day 1. So, you are already late again and if you think about 4, it will be σP_i will be is equal to 37 plus what you called as 4 will be 28 ok. So, that will be 65; 65 and 4 was supposed to be delivered on 37th day. So, you are late here.

So, by the second sequence position itself your solution is not very good you are already late big time right. Now, same way if you can think about it is so some people will say fine then there is no instead of point in going here. Let me see whether there is any other scenario where it is better. So, you can try the next one the second one and let see for an example I take the second sequence position I have 3 jobs possible here.

So, I put it as 1, 3 and 4 and if you think about it the time taken for this is 27. So, the σP_i for the first branch is it is 27 plus 1 is 27. So, that will be 64 the time to deliver is 36 it is late ok. So, then 3 will be it is 27 and 3 is time is 1. So, 28 and it has to be three has to be delivered in due date on 1, it is still late 27 plus 4 so that will be 4 is 28.

So, this case will be σp_i will be 27 plus 28 so that is 15, 2 plus 2 4 5 55. So, in this case it is a 55 and it supposed to be delivered on 37, it is still late. So, you will like fine here also by the second sequence position everybody is too late.

So, now let us look at the third one we can say is that there are three places that we can do at this point so 1, 2 and 3 so I can put 1, 2, 4 ok. So, these are the three options I have and this one if you think about is three the number of the processing time is 1. So, there is no lateness here and so then the next job is 1, we are taken and that is 37 right.

So, 37 plus 1 is 38 and it is supposed to be delivered on 49. So, no lateness l is equal to 0. If I do 2 then 2 will be 1 plus 27 that is 28 it is supposed to be delivered on 36, the lateness is equal to 0 and 4 is 28, so that is 29 ok. So, then the due date is to be 37, so the lateness is equal to 0. So, now here is a scenario where the second sequence position none of the jobs are late. So, which means now this is compared to these 3 cases I can say that this is a promising subclass.

So, compared to these guys these 2, 1 the subclass the first subclass and the second subclass this subclass is better. Because here all the jobs are late by the second sequence position and here none of the jobs are late by the second sequence position. So, then I can do is from here I can expand the third sequence position done. So, in this case it will be 2 sequence positions will be used in this case, that will be I am just going to expand it for the time being just to expand. So, this will be 3 and 1 are used.

So, it will be 2, 1, 4 or the 2 possible options in this case right so then if it is 2. So, this is 38 was a total processing time. So, far so the σP_i for us is 38 and for 2 the time is 27. So, 38 plus 27 so it is 8 plus 8 16, so it is 5 3 plus 2 is 5 plus 1; 6. So, this is 65 and 2 is supposed to be delivered by 36. So, you are late here and then with 4 you can find out 4 is 28.

So, σP_i is equal to 38 plus 28, so 16 and 3 plus so 3 plus 2 is 5; 5 plus 1 is 6. So, 66 and 4 is to be delivered at the time of 37, so this is late ok. So, both the jobs are late in the sequence position. So, now if you look at the next ones and you can say that this 32. So, in the next option is you have is 1 and 4 ok. So, these two guys will have that and then you expand it and then there is four case you will have 1 and 2 right like this.

And then from here then you calculate which is the lowest lateness and then from there you can decide which lateness or so far you might be promising and continuing that. And then wherever you find the lateness. So, this process what we are doing below this is called branching. Expanding the solution to the branches of the trees is what we called as the branching. And then let us say for example, I just let me take one example in this case I have already let us say 28 is here and then with 1 the it is 37.

So, let us take a one example is that here, this is let us say a 4 and I got that is 65. So, σP_i in our case is 65 plus the processing time of 4 is 28 so 28 that so it is 93. So, let us take it the total processing time as 93 and let us say the sum of the lateness in this

particular case is let us say you get a value of let us say for example, take it as 30 ok. So, the l value is 30.

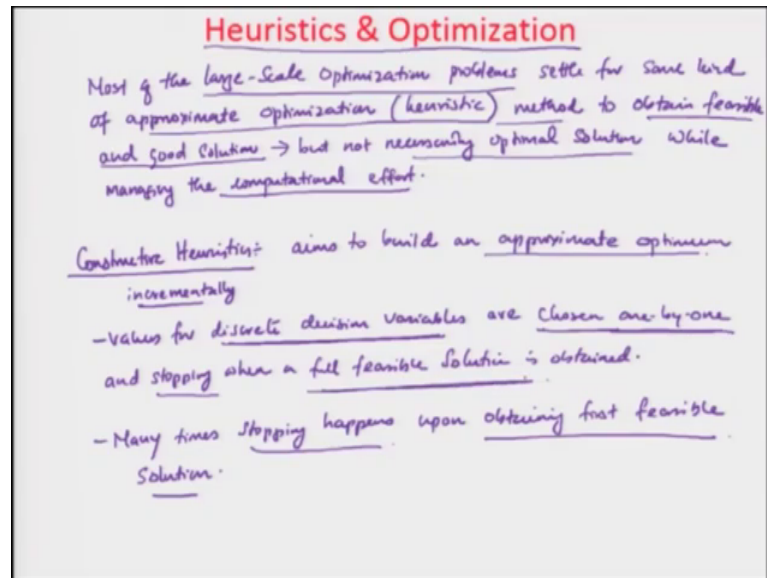
So, let us say you get the for this particular branch let us say for the l value here is 31, then what happens is you know that going below this branch is not going to help ok, because it is not going to get you any better solution. So, do not enumerate called bounding ok. You are terminating or stopping or cutting the search ok. So, there is other thing also is called bound.

So, I would not call it as bound bounding I will call it as pruning ok. What I am using is I am using this using this as the bound ok. So, I am saying that with this bound this l equal to 30 whatever I calculated I can basically stop all those branches of tress I. I can stop all those enumerations, where say look there is no need I need I do not need to expand this because my solution is not going to be better.

So, using the bound so why do you use bound? Bounds are used to reduce the number of leaves these are the leaves evaluated in a tree ok. So, this approach what I am just talking to you as of now is, what we called as a branch and bound algorithm ok.

You branch on different branches like you branching on a tree and then you use a bound to ensure that you do not evaluate all possible options in this particular example. So, I hope you understand how I am how this methods works that different implementations of these methods by different people the different variations of this also, but the fundamental idea is somewhat like this fine.

(Refer Slide Time: 48:08)



So, now we get into the last possible options which is called as a heuristics and optimization. So, the heuristics and optimization is also another important aspect that we need to consider in this case ok. So, one of the main thing that you need to understand is most of the large scale most of the large scale optimization problem, most of the large scale optimization problems settle for some kind of approximation approximate optimization approximate optimization.

You are trying to look for some approximation which we typically called as heuristic ok. Some approximate optimization method to obtain feasible and good solution ok. Some people also call, but not necessarily optimal solution. So, what you have trying while managing the computational effort computational effort ok. So, what you are trying to say is that why when you are looking at a large scale optimization problem.

So, then you set for some you try to tend to use more of an approximate optimization or a heuristic method for getting feasible and good solutions your aim is to obtain feasible and good solutions. So, when you do that what you are trying to do is you are sacrificing you are with the scenario, where you are not going to get an optimal solution, but there computational effort can be managed.

So, that is the idea and this one. So, the one thing that you need to learn about this point is called constructive heuristics. So, constructive heuristics is that aims to build an approximate optimum approximate optimum incrementally. So, step by step you make an

approximate optimum ok, an approximate optimum is created one step at a time. So, how do you do values for discrete decision variables discrete decision variables are chosen one by one.

So, you choose this variable values one by one chosen one by one. The discrete decision variable values are chosen one by one and stopping when a full feasible solution is obtained ok. As long as you have a full feasible solution and as soon as you get it is stop it ok. Many times many algorithms many times stopping happens stopping happens upon obtaining first feasible solution. So, many a times as soon as you get your first feasible solution, then you stop ok. Sometimes it can happen in single pass and then some people call it as single pass one pass heuristics and those kind of cell ok.

So, there many names and other terminologies of used in this particular scenario, but the idea is that you basically choose the values of discrete decision variables. You choose them one by one and stopping stop it when as soon as you get a full feasible solution right all right. So, this method this approach is called as a optimization heuristics in this or how do you use heuristics for optimization. You will actually see a simple algorithm and probably see an example so.

Thank you.