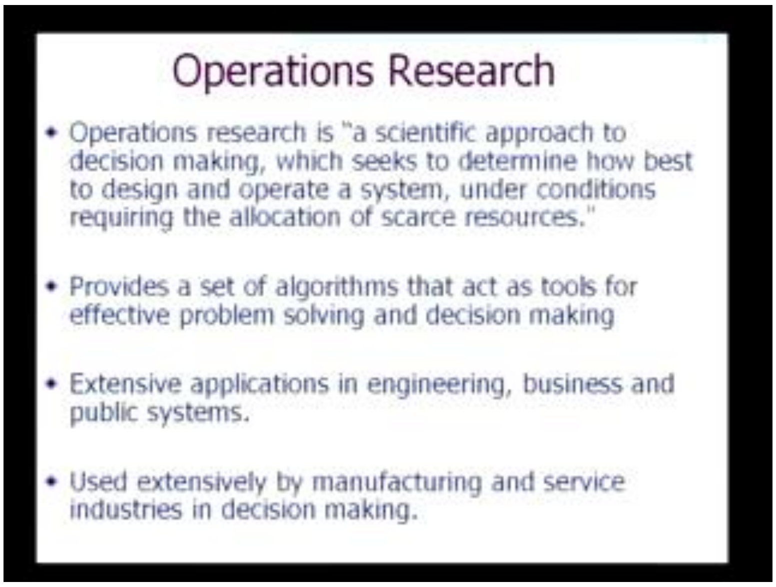
Advanced Operation Research
Prof. G. Srinivasan
Department of Management Studies
Indian Institute of Technology, Madras
Lecture No. # 01
Introduction and Linear Programming

We begin this lecture series on Advanced Operations Research. We earlier had a lecture series on Fundamentals of Operations Research where we introduced the basic topics in Operations Research. The topics that we saw earlier were linear programming, transportation and assignment problems, dynamic programming and inventory. Now we will look at some additional topics in the Advanced Operations Research course. Before we get into these topics let us first try and define Operations Research and look at some areas where this field has its applications.

(Refer Slide Time: 00:00:56 min)



Operations Research

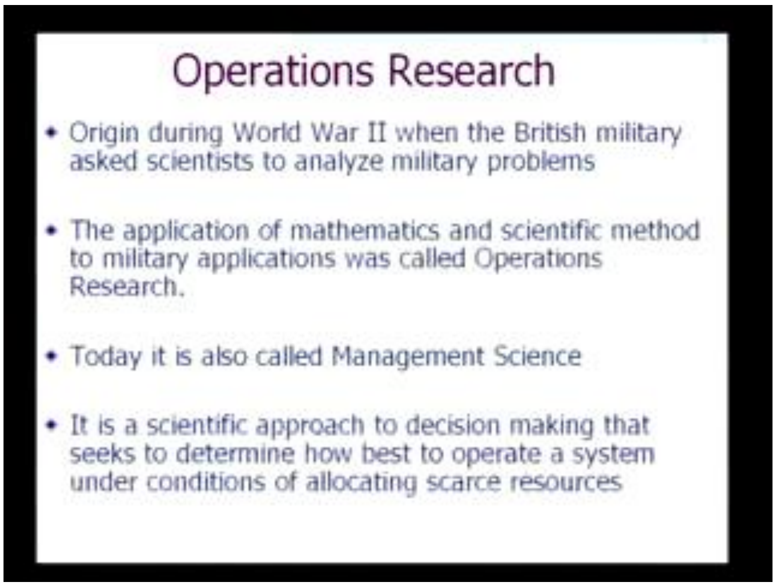
- Operations research is "a scientific approach to decision making, which seeks to determine how best to design and operate a system, under conditions requiring the allocation of scarce resources."
- Provides a set of algorithms that act as tools for effective problem solving and decision making
- Extensive applications in engineering, business and public systems.
- Used extensively by manufacturing and service industries in decision making.

Operations Research can be defined as a scientific approach to decision making which seeks to determine how best to design and operate a system under conditions requiring the allocation of scarce resources. The most important thing in this definition is the allocation of scarce resources for effective decision making. Invariably the resources are scarce and it is necessary to use these scarce resources efficiently so that the decision making is made possible.

Operations Research also provides a set of algorithms that acts as tools for effective problem solving and decision making. So essentially it is a set of algorithms and techniques which are problem solving techniques in nature which can be used in the decision making process.

As a field Operations Research has extensive applications in engineering, business and public systems. Right from solving smaller problems or smaller sized problems in the shop floor to large problems in public systems. Operations Research tools have found applications in these areas. These areas listed here which is engineering, business and public systems are only indicative and not exhaustive, Operations Research also has applications in areas other than these. OR as it is called is used extensively by the manufacturing and service industries in their decision making.

(Refer Slide Time: 00:02:44 min)

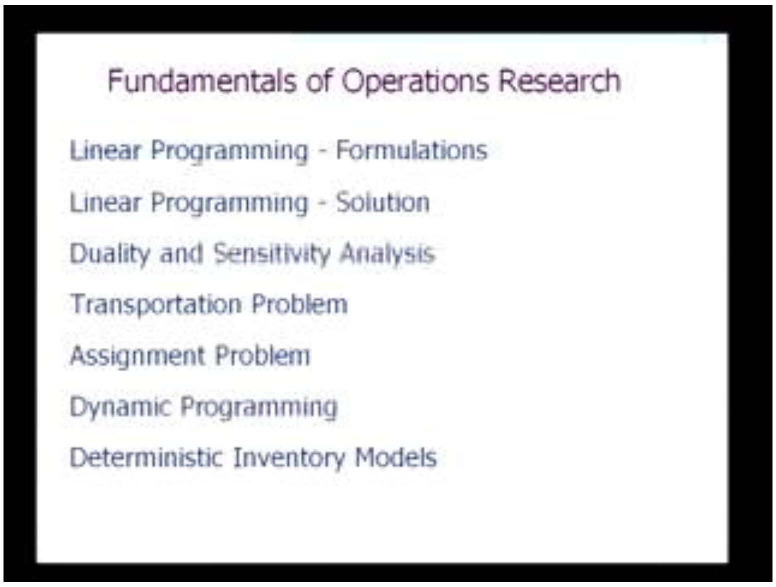


Operations Research

- Origin during World War II when the British military asked scientists to analyze military problems
- The application of mathematics and scientific method to military applications was called Operations Research.
- Today it is also called Management Science
- It is a scientific approach to decision making that seeks to determine how best to operate a system under conditions of allocating scarce resources

Now how did this field of operation research start and when did it start?
The origin of Operations Research can be traced to World War II where the British military asked scientists to analyze military problems. So what started as an application of scientific method to military is now called Operations Research. Over the years it is also called as management science and it can be also defined as a scientific approach to decision making that seeks to determine how best to operate a system under conditions of allocating scarce resources.

(Refer Slide Time: 00:03:20 min)



Fundamentals of Operations Research

Linear Programming - Formulations
Linear Programming - Solution
Duality and Sensitivity Analysis
Transportation Problem
Assignment Problem
Dynamic Programming
Deterministic Inventory Models

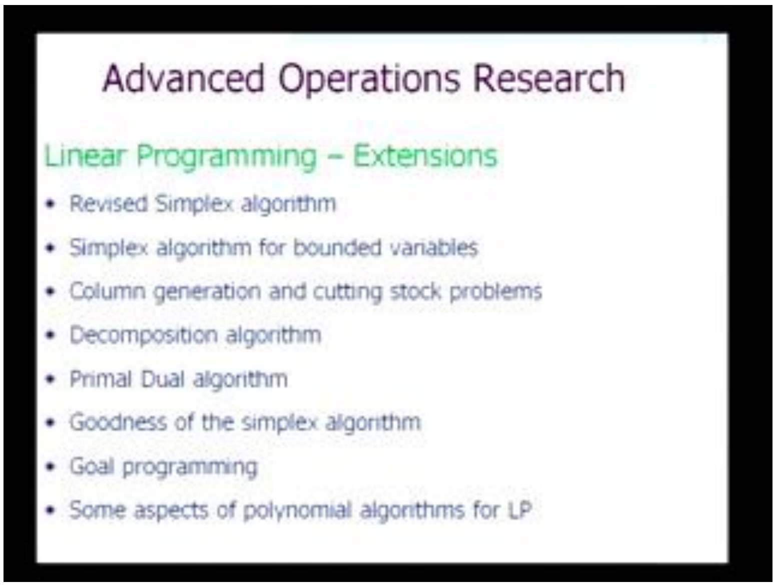
In the earlier course which was titled Fundamentals of Operations Research we looked at these seven topics;

Linear Programming - Formulations
Linear Programming – Solutions
Duality and Sensitivity Analysis
Transportation Problem
Assignment Problem
Dynamic Programming
Deterministic Inventory Models

Therefore we have linear programming – formulations, linear programming – solutions largely concentrating on the simplex algorithm beginning with the graphical method and algebraic method and then moving onto the simplex algorithm. Then we looked at duality and sensitivity analysis. Duality talks about the existence of a dual problem to a given linear programming problem. The given problem is called the primal and the new problem is called the dual. So we looked at all aspects of duality, the duality theorems, the weak duality theorem, the optimality criterion theorem, the main duality theorem, complimentary slackness theorem and conditions and sensitivity analysis. We also looked at the transportation and assignment problems which are essentially network problems but these are problems that can be solved using specialized algorithms that come out of principles of linear programming. Then we looked at a completely new approach to solving problems which is dynamic programming which essentially exploited the stage state characteristics of the problem.

So any problem to which stage wise decision making is possible where decision making at a stage can be related to the best decision in the earlier stage one could use dynamic programming to solve. Towards the end of the earlier course we saw some deterministic inventory models which talk about applications of OR techniques to solving problems in inventory and materials management.

(Refer Slide Time: 00:05:10 min)



Advanced Operations Research

Linear Programming – Extensions

- Revised Simplex algorithm
- Simplex algorithm for bounded variables
- Column generation and cutting stock problems
- Decomposition algorithm
- Primal Dual algorithm
- Goodness of the simplex algorithm
- Goal programming
- Some aspects of polynomial algorithms for LP

Now in this lecture series which is called Advanced Operations Research we are going to see a variety of topics. We begin with what is called extensions to linear programming. we are going to see all these topics; revised simplex algorithm to make the simplex faster than the version that we saw in the earlier course and how to adapt the simplex algorithm for bounded variables linear programming problem which means if the linear programming problem has bounds on the variables can we adapt the simplex method to solve the bounds instead of treating each bound exclusively as a constraint.

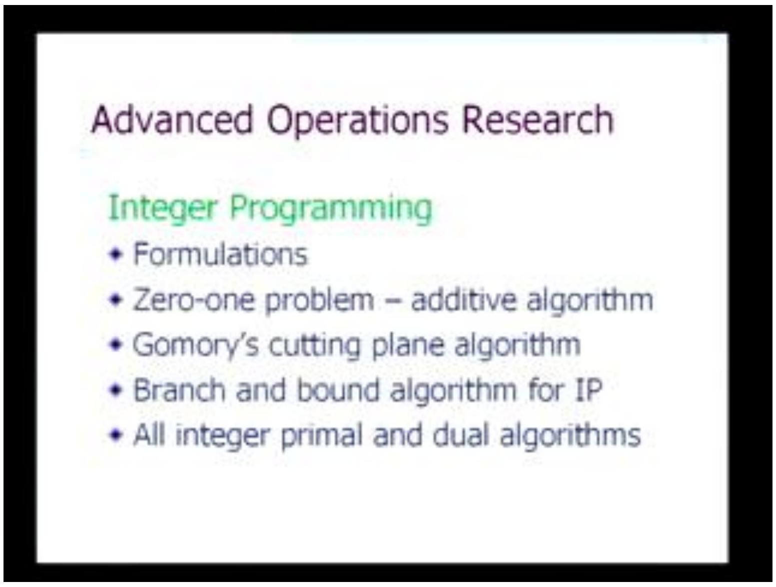
Then we will look at column generation and cutting stock problems which is one of the applications where we are going to apply these OR tools which is particularly a form of linear programming to optimize the cutting stock patterns. Then we will look at the decomposition algorithm where a large linear programming problem can be decomposed into smaller problems and by solving the smaller problems individually and by linking the solutions we can find the solution to the larger problem quickly. After this we will look at the primal dual algorithm which is essentially a kind of a dual algorithm where the dual feasibility is maintained and when the primal becomes feasible it becomes optimum.

In the earlier course we have seen some aspects of the primal dual algorithm. The Hungarian algorithm which is used to solve the assignment problem is an example of the primal dual algorithm. Then we start looking at the goodness of the simplex algorithm. How good is the simplex algorithm to solve a linear programming problem, what is its complexity, what kind of computing times it will take if the problem size increases and so on?

We will also look at goal programming. This is a way by which we address multiple objectives in linear programming. The standard linear programming problem has only one objective function which is linear. If we have more than one objective function how do we solve such linear programming problems? Do we convert them into a single objective and solve it using the simplex algorithm or do we solve each objective one after another, do we rank them and so on. Thus all these questions are addressed in goal programming.

And towards the end of this module called extensions to linear programming we will look at some aspects of polynomial algorithms for linear programming. We require algorithms with that run in polynomial time so that large sized problems can be solved effectively. Now we want to check whether the simplex algorithm has a polynomial time complexity or what are the issues associated with polynomial algorithms for linear programming. Now these are the topics that we will see in the first part which is called linear programming extensions.

(Refer Slide Time: 00:08:25 min)


Advanced Operations Research

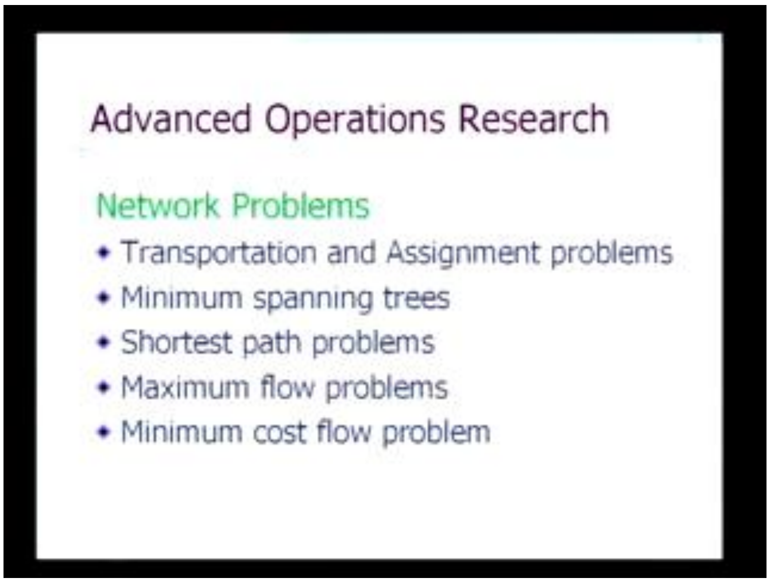
Integer Programming
- Formulations
- Zero-one problem – additive algorithm
- Gomory's cutting plane algorithm
- Branch and bound algorithm for IP
- All integer primal and dual algorithms

Then we move to integer programming. Integer programming becomes important when the decision variables take integer values. In a linear programming problem the decision variables take continuous values whereas in integer programming they can take integer values or they can take binary which means they take either 0 or 1 value. We have already seen some aspects of this. The assignment problem can be seen as a 0 1 integer programming problem even though it can be solved as a LP and therefore it is solved as a LP. In this module we will look at all these. We will look at formulations of integer programming problems. Many real life problems are formulated as integer programming problems. So we will see some examples of formulations.

We will then go on to solve zero-one integer programming problems through an algorithm called the additive algorithm which is an implicit enumeration algorithm. Then we will look at solving all integer problems which means all the variables in the formulation are constrained or restricted to be integers. So we will look at the Gomory's cutting plane algorithm which is a very popular and a well known algorithm to solve integer programming. We will also look at branch and bound algorithm for integer programming and we will also look at what are called all integer primal and dual algorithms. The all integer primal and dual algorithms use the integer property of the coefficients and in all the iterations the coefficients remain as integers.

So in terms of computational simplicity all integer primal and dual algorithms score over the Gomory's cutting plane algorithm where you may have variables taking fractional values in the intermediate iterations. Though from computational point of view Gomory's cutting plane algorithm is preferred to all integer primal and dual algorithms. Now this would complete the second module or second part of this course called Advanced Operations Research.

(Refer Slide Time: 00:10:48 min)



Advanced Operations Research

Network Problems

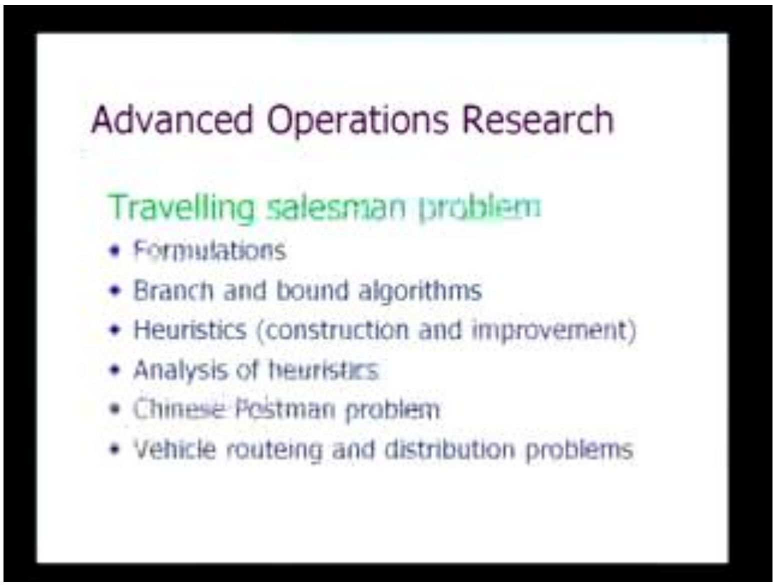
- Transportation and Assignment problems
- Minimum spanning trees
- Shortest path problems
- Maximum flow problems
- Minimum cost flow problem

After that we will look at what are called network problems. Network problems are typically zero-one problems or integer programming problems but have what is called a network structure. Now because of this network structure you can solve them as linear programming problems and yet get integer valued solutions.

First examples of network problems are transportation and assignment problems that we have seen in the earlier course as part of linear programming. Now all the network problems have something called the unimodularity property which means that if the problem is of the form AX equal to b the constraint coefficient matrix is unimodular which means that the LP solution will give integer values.

Now this property is being exploited to solve all these network problems; transportation, assignment, minimum spanning trees, shortest path problem, maximum flow problem and minimum cost flow problem. Now the unimodularity property based on which the LP solution will be integer valued is used to develop and derive special algorithms which are based on principles of linear programming but are slightly different from simplex and these algorithms are used to solve these problems. So we will look at an algorithm or two for each one of these problems.

(Refer Slide Time: 00:12:23 min)



Advanced Operations Research

Travelling salesman problem

- Formulations
- Branch and bound algorithms
- Heuristics (construction and improvement)
- Analysis of heuristics
- Chinese Postman problem
- Vehicle routeing and distribution problems

After this we will look at the travelling salesman problem which is a very well known problem in the field of Operations Research. It is also a very widely researched problem and is usually it comes under the category of easy to state and difficult to solve problem. The problem is well known; given a sales person who has to visit n cities starting from his starting point how does this person visit each city once and only once and comes back to the starting point traveling minimum distance possible. This problem has a history and this problem has been researched extensively over the last many years.

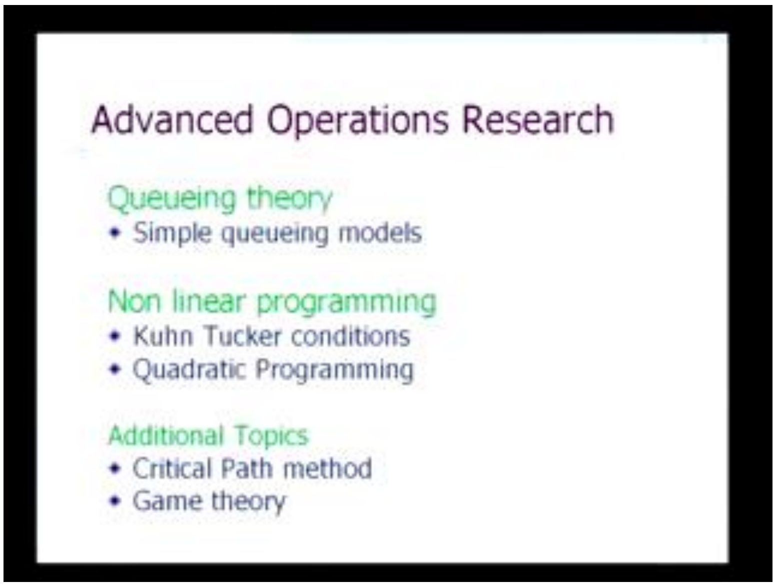
Now in the travelling salesman problem topic we will look at all these, how this problem is formulated as an OR problem, what are the algorithms particularly branch and bound algorithms. So we will be looking at more than one type of a branch and bound algorithm to solve this problem. We will also be looking at heuristics. Heuristic algorithms are algorithms that do not essentially guarantee the optimal solution but they provide very good solutions very quickly. The advantage is in the fact that computationally the heuristics take very little time and also provide solutions which are quite close to the optimal solution.

Travelling salesman problem is a minimization problem because we try to minimize the total distance traveled and the heuristic solutions always give a value which could be the optimum or higher than the optimum. So they act as upper bounds to the optimum solution to a minimization problem.

We will also see some aspects of analysis of heuristics, how good the heuristic is, what is the worst case performance of the heuristic. So we will look at those aspects by considering two very well known heuristics to the TSP which is called twice around the tree heuristic as well as a heuristic based on matching. Both these also involve the minimum spanning tree which we would have covered by then under the network problem. There is a slight variation of the TSP which is called the Chinese postman problem. in some sense it's not a variation as such but it's a problem that is very close to the TSP while the TSP talks about visiting every vertex or a node once and only once and coming back to the starting point the Chinese postman problem talks about visiting each arc or edge of a network once and only once if possible and coming back to the starting point. So in that sense it is related to the TSP.

We will see some aspects of the Chinese postman problem in this course after which we will look at vehicle routing and distribution problems which are typical extensions of the travelling salesman problem. In a vehicle routing problem we talk about multiple vehicles originating from a point picking up or dropping people or material or transport goods and come back to the starting point. So a vehicle routing problem can be seen as multiple TSPs and we also look at some more distribution problems where there could be a restriction on the size of the vehicles, there could be a restriction on the number of people that the vehicle could carry and so on. So we will look at algorithms to solve vehicle routing and distribution problems. After this we will look at a bit of queueing theory a bit of nonlinear programming and a couple of additional topics.

(Refer Slide Time: 00:16:11 min)



Advanced Operations Research

Queueing theory
- Simple queueing models

Non linear programming
- Kuhn Tucker conditions
- Quadratic Programming

Additional Topics
- Critical Path method
- Game theory

Queueing theory is perhaps the only topic in this course which is not deterministic in nature. Every other topic that we are going to see comes under deterministic OR where the problem parameters are deterministic known a priori. Queueing theory is perhaps the only topic in this course where we will look at some nondeterministic some probabilistic models.

So here we will look at arrivals and service patterns, we will look at single server and multiple server systems, we will look at infinite queue length and finite queue length models and so on. so here we would assume that the arrivals will follow the Poisson distribution, the service times are exponentially distributed and within this context we will see how the queuing systems are performing in terms of expected number of people in the system, expected number of people in the queue, the average or expected waiting time in the system and expected waiting time in the queue. So all these four parameters will be measured and formulae or derivations to get expressions for these parameters will be shown and derived in this course.

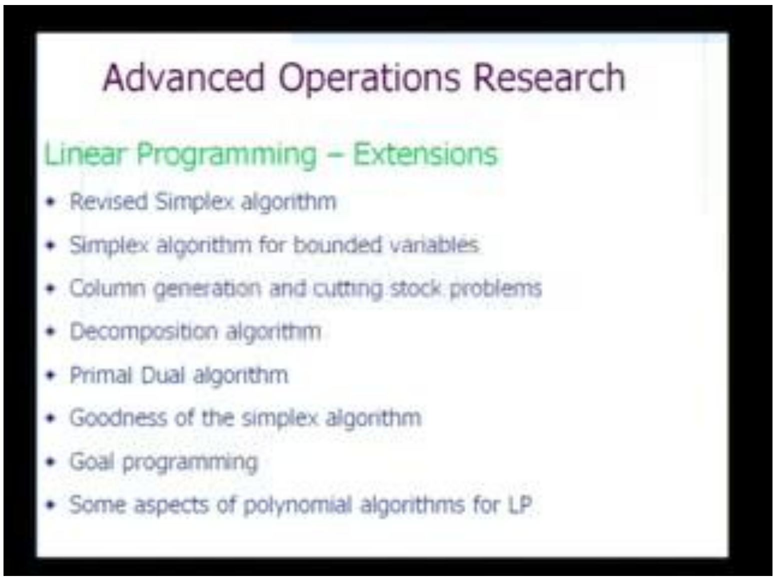
We will then look at nonlinear programming and nonlinear optimization. We would also look at the Kuhn Tucker conditions which are the conditions that are used extensively to solve nonlinear programming problems. We will also concentrate on one specific aspect of nonlinear programming which is called quadratic programming where the objective function is quadratic in nature, the constraints are linear and there is an explicit non negativity restriction on the variables. We will also show the algorithm to solve the quadratic programming problem which involves linear programming principles.

There are two other additional topics that we will look at one is called CPM which is the Critical Path Method and from an OR point of view this can be looked upon as a network problem where we find the longest path of a certain type of network. In general the longest path on a network is a hard problem to solve but for certain types of networks which are there in the CPM we will see that one can use a very efficient algorithm to find out the longest path.

So in some sense CPM can be seen as part of the network models topic but CPM is a specialized application because of its extensive use in project management. We will also see some aspects of game theory some simple things such as saddle point, graphical solution to a two person zero sum game and solution to two persons zero sum game using linear programming.

In the earlier course on Fundamentals of Operations Research we have actually formulated the game theory problem as a linear programming problem. Now here we will also show the primal dual relationship between A's game and B's game if A and B are the two persons who are actually playing this zero sum game. So this is in some sense the overview of this course and what we are going to see in the Advanced Operations Research course.

(Refer Slide Time: 19:46)



Advanced Operations Research

Linear Programming – Extensions

- Revised Simplex algorithm
- Simplex algorithm for bounded variables
- Column generation and cutting stock problems
- Decomposition algorithm
- Primal Dual algorithm
- Goodness of the simplex algorithm
- Goal programming
- Some aspects of polynomial algorithms for LP

So quickly coming back to the starting point of this course we will now continue and begin this course formally by looking at the revised simplex algorithm. So we start with the revised simplex algorithm. Now, as the name suggests the revised simplex algorithm is a form of the simplex algorithm. So, before we get into the revised simplex algorithm let us spend a little bit of time on trying to go through the basics of the simplex algorithm that we have seen in the earlier course.

Now in the earlier course we have seen that the simplex algorithm provides the optimal solution to linear programming problems. We have also seen that the optimum solution to a linear programming problem is a corner point solution and therefore the simplex algorithm is also tailored in that manner to find out the corner point optimal solution.

So the simplex algorithm starts with a corner point and then moves along the boundary of the feasible region from one corner point to another till it actually reaches the corner point which is the optimum solution to the linear programming problem.

Now what does a simplex algorithm do? Now, if we consider a linear programming problem with n variables and m constrains. Usually n is greater than m the number of variables is greater than the number of constraints.

We have all we have already seen in the earlier course that these variables can be of several types, these variables can be the problem decision variables, they could also be slack variables where a slack variable is introduced to convert an inequality into an equation. And within these slack variables it can be a positive slack or a negative slack. The negative slack is also called the surplus variable, it could also be artificial variables where these artificial variables are introduced so that we are able to get an identity matrix in the simplex algorithm. So the variables are of three types.

The other way of saying that these variables from a simplex point of view can be a greater than or equal to variable, can be an equation, can also be a less than or equal to type variable. Linear programming problem also has constraints and there are three types of constrains. The constraint is a greater than or equal to type constraint, it can be a less than or equal to type constraint and it can be an equation. So there are three types of variables in the linear programming problem, there are three types of constraints in the linear programming problem and the linear programming problem also has an objective function which can be a maximization objective function or a minimization objective function. We have looked at three aspects of the simplex algorithm in the earlier course and these three aspects are called initialization, iteration and termination.

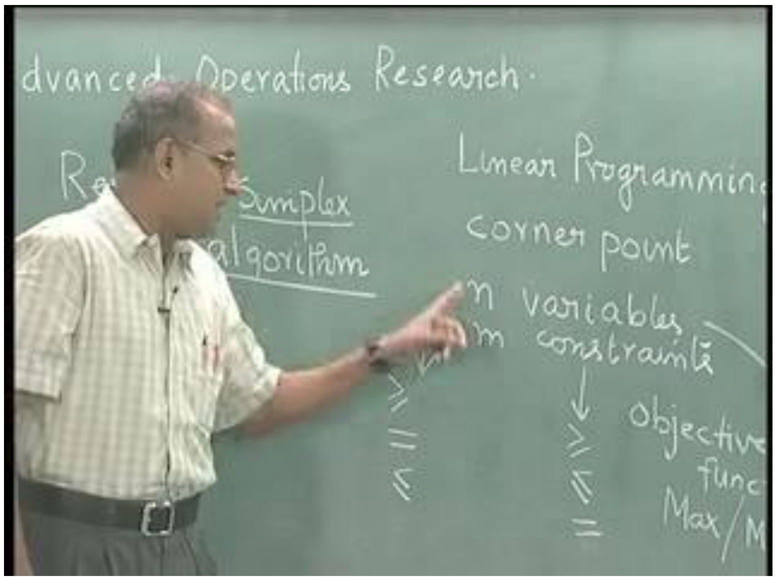
(Refer Slide Time: 00:24:10 min)

Now in the initialization we try and bring the given linear programming problem into a format where the simplex method can readily be used to solve. So the initialization essentially tries to bring it into what is called the canonical form wherein an identity matrix can be easily read from the starting solution.

In the iteration at the initialization stage we assume that we have a basic feasible solution and in the simplex algorithm this basic feasible solution is a corner point solution and in the simplex algorithm we move from one basic feasible solution to another from one corner point to another and then at the end the algorithm terminates. In each iteration simplex evaluates a different basic feasible solution.

Now what is this basic feasible solution? We have said that there are n variables and m constraints and let’s assume that these n variables include all the types of variables which means al the constraints have now been converted to equations and if required all the artificial variables have been added so that n is clearly greater than m. Now, once we have more variables than constraints we also know that if we want to solve equations we can solve only for as many variables as the number of constraints that we have. So we can solve only for m variables in any iteration.

(Refer Slide Time: 26:22)



So out of these n variables that we have n is greater than m we choose m variables to solve and the remaining n – m variables will be called non basic variables which are fixed at 0 so that we solve for the rest of the variables which are the basic variables. Such a solution is called a basic feasible solution and this basic feasible solution will be a corner point solution. So, in the initialization we ensure that we get a basic feasible solution. In iteration we move from one corner point solution to another with progressively better values of the objective function.

If the problem is a maximization problem we not only ensure that every solution evaluated is basic feasible but we also ensure that the objective function does not decrease (or increases) in every iteration. At the end the algorithm will terminate either when it reaches the optimal solution or some other things happen.

Therefore in order to now understand the termination condition we should also go back and review the fundamental theorem of linear programming which says that every linear programming is either feasible or unbounded or infeasible. So every linear programming is feasible or unbounded or infeasible.

Now the fundamental theorem also says that if it is feasible it has a basic feasible solution and in this case it is expected to terminate with the optimum solution. So, simplex will terminate either with the optimum solution or by indicating an unbounded solution or by indicating an infeasible solution. Within this optimum case you will have a unique optimum or you will have an alternate optimum.

In the unbounded we have defined the termination condition for unbounded and we have defined the termination condition for infeasible in the earlier course. Let us quickly take stock of that. Now in the format that we have used we will find that for a maximization problem all $C_j – Z_j$ is less than or equal to 0 we say that the optimum is reached and once the optimum is reached if we have another nonbasic variable with zero then it indicates alternate optimum. These are the two cases unique optimum and alternate optimum in this case.

If we have an entering variable which means if we have a variable with a positive $C_j – Z_j$ it can enter and we are unable to find a leaving variable then it indicates unboundedness and the linear programming problem is unbounded. And in the infeasible case there is one way of showing infeasibility that the optimality condition is satisfied which means all $C_j – Z_j$ is less than or equal to 0. However, you still have an artificial variable lying in the basis with a strictly positive value. If that happens then we say it is infeasible.

Now if all $C_j – Z_j$ is less than or equal to 0 and if we find an artificial variable with a value equal to 0 then we may say that the system of equations is linearly dependent. So simplex has its own way of showing even that the system of equations is linearly dependent and therefore simplex can be used to solve a system of equations. But coming back to the point that we are discussing the three aspects are feasible, unbounded and infeasible and these three define the termination condition for the simplex algorithm. In addition we may say that if simplex does not terminate in spite of all these then simplex does what is called cycling - simplex cycles.

We briefly mentioned about cycling in the earlier course. Cycling is when you start with a set of basic variables which is called a basis and after several iterations you come back to the same basis. Now such a thing is called cycling and there have been some instances where simplex had actually cycled.

Cycling is a limitation of the simplex algorithm and it has nothing to do with the linear programming problem. People over the period of time have also found out ways by which cycling can be eliminated. Cycling can be eliminated by the use of what is called Lexicographic rule and it can also be eliminated by the use of what is called the Bland’s rule or the smallest subscript rule.

Hence if there is a tie for a leaving variable and if there is a tie for the entering variable we will always choose consistently the variable that has the smallest subscript and that is enough to prevent cycling. There is a proof that the smallest subscript rule is indeed optimum. The Lexicographic rule involves a little more computation and we would look at the lexicographically smallest variable to be the choice for the leaving variable or the entering the variable if there is a tie so either of these rules can be used to prevent the cycling of the simplex algorithm if it really cycles.

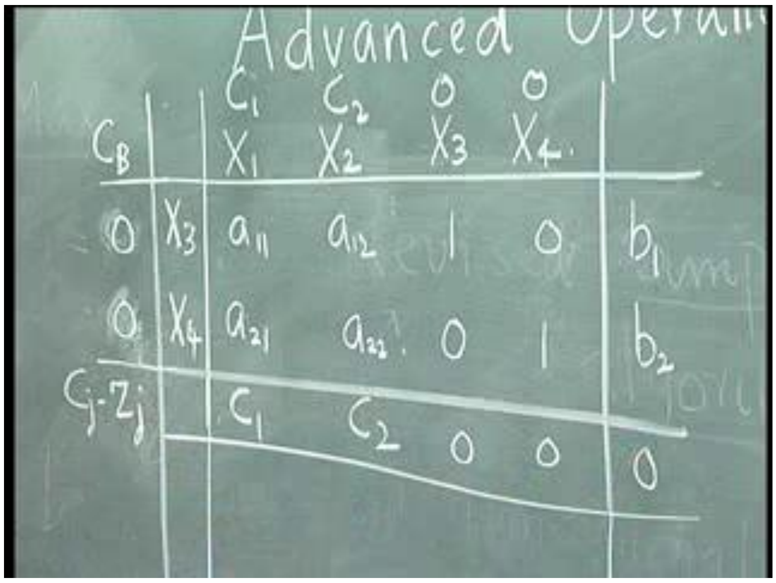
Incidentally we still haven’t come across any problem that has been formulated out of a real life situation which actually cycles. Most problems that are available in the literature which cycle are problems that have been created by the researches themselves.

Now, having seen so much of linear programming problems and their simplex algorithm we will come back to where we wish to start which is to look at the revised simplex algorithm. Now what is this revised simplex algorithm and why do we need the revised simplex algorithm?

Before we get into that let us look at two aspects of the speed or goodness of the simplex. The two aspects are; time taken by the simplex algorithm per iteration and the second is the number of iterations. So if we want the simplex algorithm to be a very fast algorithm then simplex has to be good on these two dimensions. The time it takes per iteration has to be as small as possible and the number of iterations it takes before it terminates should be as small as possible so that simplex algorithm is able to give the optimal solution very quickly.

Now let us see what actually happens in these two aspects of the simplex algorithm and then move on to go to this revised simplex which essentially tries to minimize the first one. Of course there are some advantages of the revised simplex algorithm as well. So we will first look at the number of iterations and then we will look at the time per iteration and then proceed towards the revised simplex algorithm.

(Refer Slide Time: 00:34:47 min)



Advanced Operati

		C_1	C_2	0	0	
C_B		X_1	X_2	X_3	X_4	
0	X_3	a_{11}	a_{12}	1	0	b_1
0	X_4	a_{21}	a_{22}	0	1	b_2
$C_j - Z_j$		C_1	C_2	0	0	0

Now let me show the simplex table first and then address the number of iterations aspect. Now, if we take a four variable two constraint linear programming problem and call our variables as X_1 X_2 X_3 X_4 then in the simplex table convention that we have used earlier now this portion has the basic variables so let us call X_3 and X_4 as the basic variables. Now we have something called C_B which is the objective function coefficient of the basic variable. So you will have C_3 and C_4 here and in the simplex table because X_3 and X_4 are the basic variables you will find and identity matrix under these and then you will also find some coefficients which are here which you may call as some a_{11}, a_{12}, a_{21} and a_{22} we call here the objective function coefficients to be C_1 and C_2 if we assume that X_3 and X_4 are the slack variables you would start with 0 0 which means C_3 is 0 and C_4 is also 0 then we evaluate C_j –

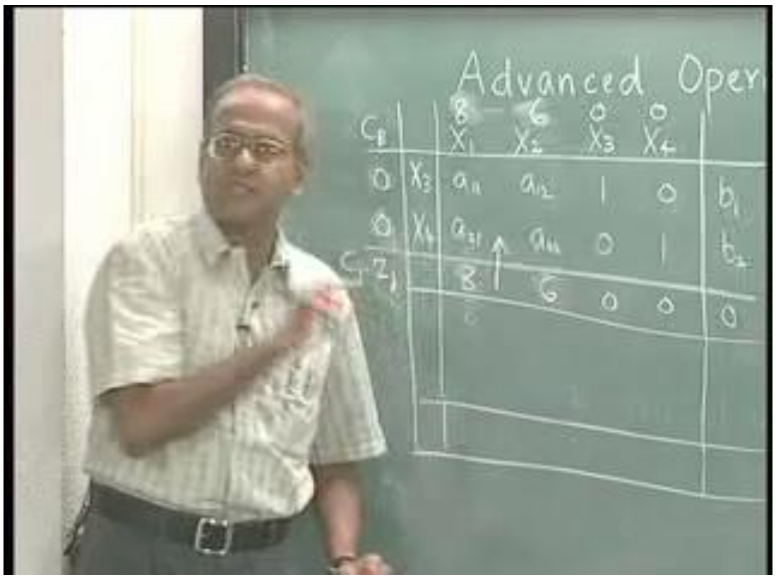
Z_j which is this minus the dot product of these two vectors (Refer Slide time: 36:24). So in this case it will be simply C_1 and C_2 because these are 0 so you will have 0 and this will be the right hand side value which would be some b_1 b_2 and this will be 0 because the right hand side will be the product sum of zero into b_1 + 0 into b_2 which will be 0.

Now this will be the optimal solution if C_1 and C_2 are less than or equal to 0. If one of them is positive or both of them are positive then one variable will enter the basis. Now that is called the entering variable. And then once a variable enters there is something called the leaving variable. For example, let us assume that C_1 is positive and for a moment C_1 is greater than C_2 so C_1 will enter based on what is called the largest coefficient. In this case X_3 and X_4 are basic so X_1 and X_2 are non-basic if more than one non basic variable has a positive $C_j - Z_j$ then the one that has the most positive value will enter based on what is called the largest coefficient rule. So let's assume that C_1 is greater than C_2 and both are greater than or both are greater than 0 therefore C_1 will enter based on the largest coefficient rule. Now this is called the entering variable. Now we need to find out a leaving variable and to do that we compute something called theta which is b_1 by this b_2 by this provided this coefficient is strictly positive. In simplex right hand sides are always greater than or equal to 0 therefore the theta value has to be strictly positive.

So in this one the theta value can be 0 or strictly positive and theta cannot be negative because when these become 0 or negative we do not evaluate the corresponding theta. We have already seen that theta represents the extent to which this variable can be increased or this variable can go up till this current basic variable becomes 0. So let's assume that all these are nonnegative they are nonzero they are strictly positive so we will evaluate b_1 by a_{11} and b_2 by a_{21} and choose the minimum of them so the minimum will leave. For example, if this is smaller, then this will leave this is called the leaving variable. When there is a tie for the leaving variable we say that there is degeneracy and there will be a 0 in the next iteration. There is also a choice for the leaving variable.

At the moment we do not have a very specific tie breaking rule for the leaving variable. But in order to safeguard ourselves against cycling it is always good to consistently use the smallest subscript rule to identify the leaving variable if there is a tie. Now the real issue comes in deciding the entering variable. We said that if more than one is positive we could enter with the most positive one. For example, if this is eight and this is six (Refer Slide Time: 40:23) then this is 8 and this is 6 now based on the largest coefficient rule this one will enter. There is nothing wrong in actually entering this as long as this is positive. But by convention we always use the largest coefficient rule.

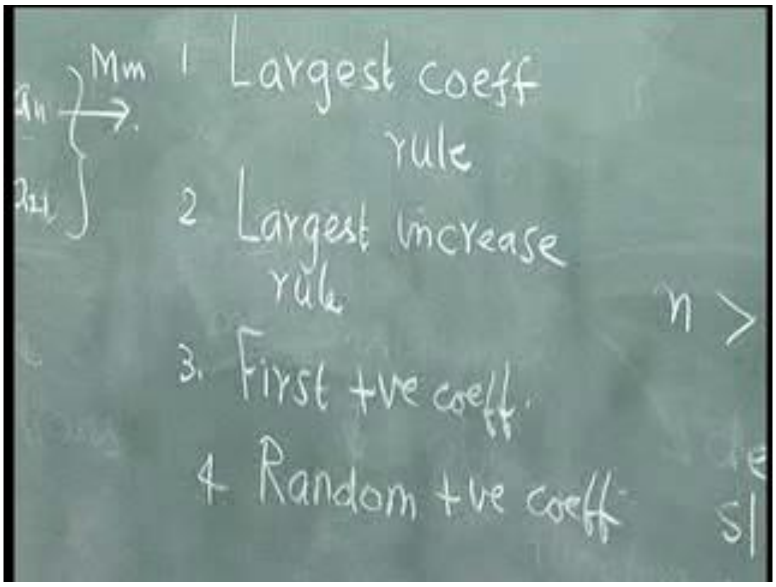
(Refer Slide Time: 40:47)



Now the question is, by a judicious choice of the entering variable can we bring down the number of iterations in simplex or can we reduce the CPU time or the computational time required for simplex?

So people started working on what are the various ways by which you can enter a variable if you have more than one variable with a positive $C_j - Z_j$. There are four very popular rules that are being used for the entering variable and these four famous rules are the largest coefficient rule, the largest increase rule, first positive coefficient rule and random positive coefficient rule.

(Refer Slide Time; 00:41:26 min)



a_{11} } $\xrightarrow{Mm}$
a_{21}

1 Largest coeff rule

2 Largest increase rule

3. First +ve coeff.

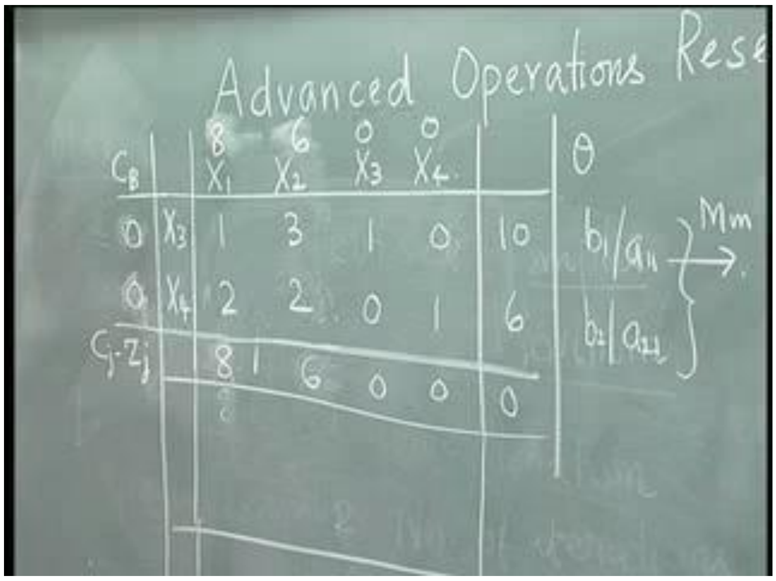
4 Random +ve coeff

n >

We have already seen the largest positive coefficient rule. If you have 8 and 6 here and both are positive so any one of them can enter, this has a larger coefficient (Refer Slide Time: 42:22) so this enters so by the largest coefficient rule or the largest $C_j - Z_j$ rule this will enter.

Now what happens if it is a largest increase rule? Now what we will have to do in the largest increase rule is let’s assume that these values are 1 and 3 2 and 2 let’s assume that this value is 10 and 6.

(Refer Slide Time: 43:04)



Advanced Operations Rese

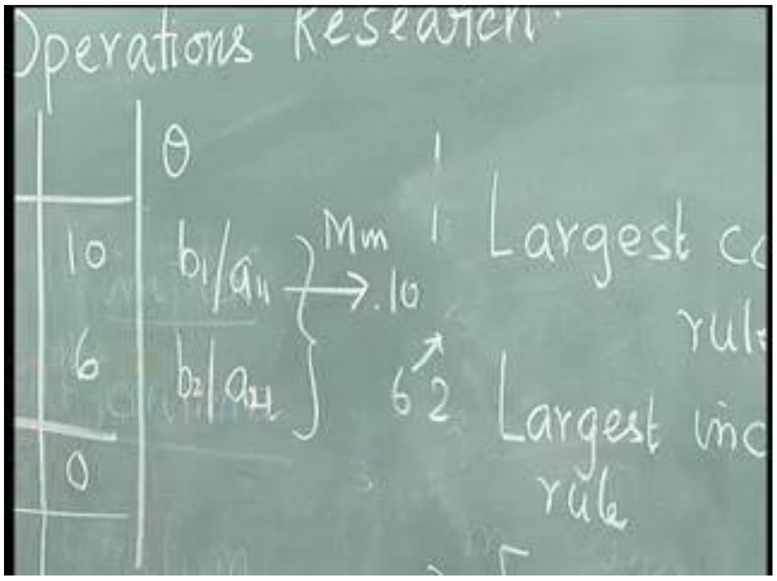
		8	6	0	0		
C_B		X_1	X_2	X_3	X_4		θ
0	X_3	1	3	1	0	10	b_1/a_{11}
0	X_4	2	2	0	1	6	b_2/a_{21}
C_j-Z_j		8	6	0	0	0	

$\} \xrightarrow{\text{Min}}$

Let’s assume this is one so this has a positive value so if we enter this then the leaving variable is ten divided by one which is 10 and 6 divided by 2 which is 3 so the minimum of that will be 3 so theta will be 3. Now in the next iteration the increase is always the product of $C_j – Z_j$ and the corresponding theta. So, if we enter this $C_j – Z_j$ is 8, theta is 3 so the product is 24.

If on the other hand we choose to enter this now the corresponding theta will be 10 divided by 3 so 10/3 will be theta so this will be 6 into 10/3 which is 20 and therefore once again this will enter based on the largest increase rule. But if on the other had this also had been 1 (Refer Slide Time: 44:26) and if this enters now theta will be the minimum of 10/1 and 6/6 which is 6 so the increase will be the product of $C_j – Z_j$ into theta which will become, here you will have 10 and 6 and this is the minimum theta so 6 into 6 = 36.

(Refer Slide Time: 44:47)



If you had entered this it will be 8 and 3 = 24 so based on the largest increase rule this variable will enter because this shows a larger increase from 0 to 36 whereas this would have increased from 0 to 24. So depending on the rule a different variable will enter and the basis can change.

The first positive $C_j – Z_j$ is a very simple rule simply look at the $C_j – Z_j$ row and identify the first positive $C_j – Z_j$. in this case this will enter because this is the first positive Cj – Zj. If for a different problem the $C_j – Z_j$ values are – 3, 1, 9, 0 0 this will enter because this is the first positive $C_j – Z_j$. Random positive $C_j – Z_j$ is there are five values randomly so pick one and if it is positive enter. so if the random number chosen is 3, then this will go if the random number chosen is one it will look at this – 3 and say that this not positive therefore choose another random number and if the random number becomes 2 then it will look at the second position that is the 1 so the second variable will enter. Like this we can start defining various entering rules so that the simplex algorithm proceeds by identifying a newer solution.

Now let us try and look at all four of them. Now this is the simplest and the easiest. From a computational view point we need this vector the $C_j – Z_j$ row and in some sense we have to identify the largest which means it is equivalent to sorting it based on decreasing value and picking the first. So the computational effort involved is simply sorting a given vector.

Now here for every positive you have to find out a theta and then multiply and then find out that variable which has a positive $C_j – Z_j$ and has the highest increase. So this requires a little more computation than this. These two also require lesser computation. Here it is much simpler, you simply look at the vector and find out the first positive value and enter that variable.

Here (Refer Slide Time: 47:25) there is a certain computation involved because you have to randomly start picking a variable, check first of all whether it is positive if so enter if not randomly pick another variable and so on. So out of all these four based on computational studies people found out that this requires fewer iterations in general but this (Refer Slide Time: 48:00) takes the least CPU time compared to all four of them.

So in terms of computational time people said this is a good rule, this is the best out of these four rules and based on this simply because we want our simplex to be quick we always try and pick the largest coefficient rule even though for other problem instances we may have a situation where other rules can do better.

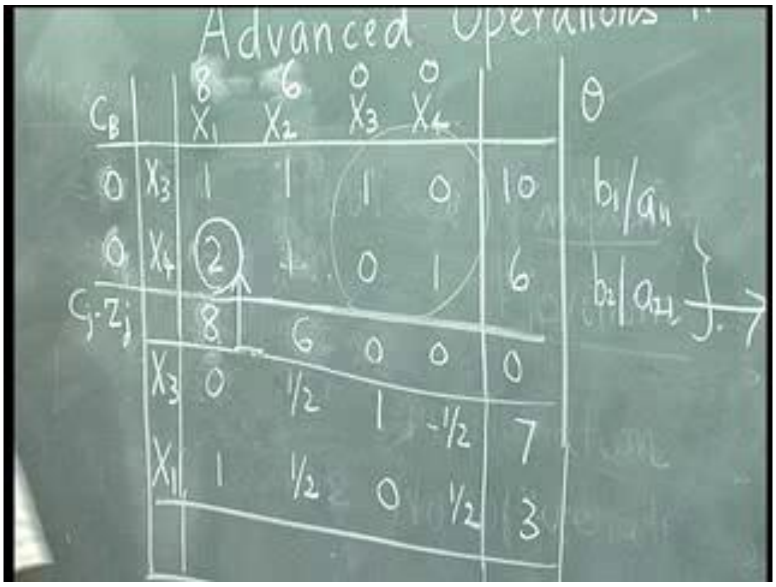
So we pick the largest coefficient rule usually in solving using the simplex algorithm simply because computational tests have shown that it takes the least amount of CPU time or least amount of computational time. So it is always possible or this always happens that we end up choosing the one which has the largest coefficient or the largest $C_j – Z_j$. The leaving variable rule is always the one with minimum theta and if there is a tie it is always good to choose the minimum subscript consistently so that we safeguard ourselves against cycling in linear programming.

So we tried to answer the first question which is what is the entering variable or how is the entering variable chosen? Entering variable is chosen based on the largest coefficient rule. Let’s go back to the two questions that we asked when we began which are the number of iterations and time taken per iteration. So in some sense we solved the number of iterations problem by looking at largest coefficient rule. One may argue that the largest increased rule actually has less iteration but then it is not only the number of iteration it is the total CPU time. And because we want the total CPU time to be minimized we prefer the largest coefficient rule in preference to the largest increase rule.

So we move to the next one which is time taken per iteration. Now, what is the time taken per iteration? Let’s assume that we have iteration like this (Refer Slide Time: 50:33) from which we are going to move to another iteration.

Now let us assume that this is the entering variable and based on these numbers this is a leaving variable. This is the entering variable you need to compute the theta 10 divided by 1 is 10 and 6 divided by 2 = 3 so the minimum theta leaves so this is the one. This is called the pivot row and this is called the pivot element.

(Refer Slide Time: 00:51:12 min)



Advanced Operations

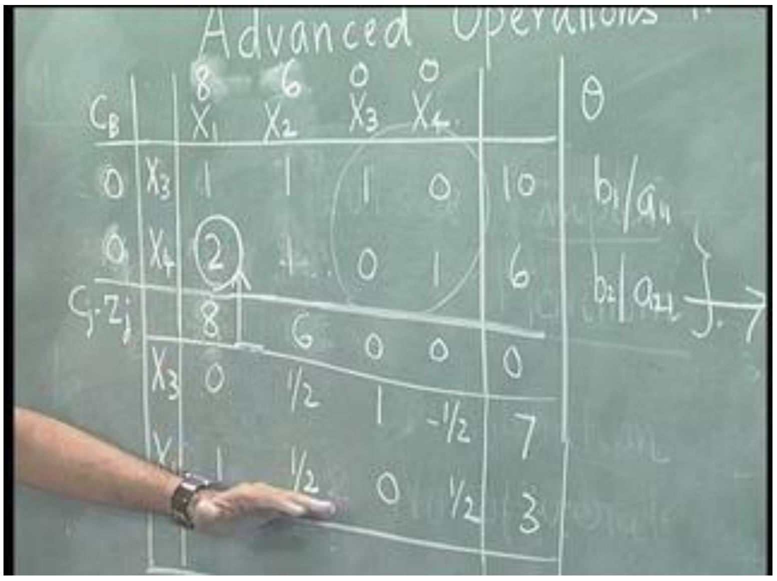
		8	6	0	0		
C_B		X_1	X_2	X_3	X_4		θ
0	X_3	1	1	1	0	10	b_1/a_{11}
0	X_4	$\boxed{2}$	1	0	1	6	b_2/a_{21}
$C_j - Z_j$		8	6	0	0	0	
	X_3	0	1/2	1	-1/2	7	
	X_1	1	1/2	0	1/2	3	

Normally in our tabular form or hand calculation of simplex we follow a set of rules which are; Now we replace this so X_3 remains, X_4 is replaced by X_1. Now when we had X_3 and X_4 as the basic variables you had the identity column somewhere here. Now in this iteration we will have to ensure that the identity column shifts to X_3 and X_1 so that the solution whatever we get here is the corresponding basic feasible solution. So we need to do row and column operations such that we retain the 1 0 here on X_3 and we have 0 1 on X_1. So the identity matrix taken as X_3 X_1 will have 1 0 here and 0 1 here.

Now we achieve this in the tabular form using a certain set of steps which is divide every element of the pivot row by the pivot element which would give us 2 divided by 2 as 1, 1 divided by 2 as ½, 0, ½ and 3.

Now to fill the rest of them what we have to do is this; I need a 0 here (Refer Slide Time: 52:29) so this minus this is 0 so 1 – 1 = 0, 1 – ½ = ½ is ½, 1 – 0 = 1, 0 – ½ = – ½ and 10 – 3 = 7. So, through these two steps the pivot row is always divided by the pivot element and for the rest of the rows you do row operations such that you get the identity matrix on this, you will automatically retain the identity matrix on this and you get the next iteration.

(Refer Slide Time: 53:03)



Advanced Operations

C_B		8	6	0	0		θ
		X_1	X_2	X_3	X_4		
0	X_3	1	1	1	0	10	b_1/a_{11}
0	X_4	$\boxed{2}$	1	0	1	6	b_2/a_{21}
$C_j - Z_j$		8	6	0	0	0	
	X_3	0	1/2	1	-1/2	7	
	X_1	1	1/2	0	1/2	3	

Now what we have done here is the famous Gauss Jordan way of solving equations. It is called the Gauss Jordan method of either solving an equation or inverting a matrix. We have already seen in the earlier course that if X_3 and X_1 are the basic variables we can go back and find out the basis matrix and you can invert it you will get exactly this. Now this is true because what we have actually done is if X_3 and X_1 are your basis so you start with 1 0 1 2 now you have got 1 0 0 1 which means you have premultiplied it by its inverse so you have premultiplied it by B inverse to get 1 0 0 1.

Now this is an identity matrix so you premultiply it by the same B inverse to get 1 0 and minus ½ ½ therefore under the original identity matrix you will have the B inverse. We have also seen that every iteration of simplex is equivalent to solving equations. Now this is equivalent of solving for something into X_1 plus something into $X_3 = 1$ and this is equivalent of solving some other thing into X_3 plus some other thing into $X_1 = 1$. So every iteration of the simplex is about solving a set of equations, it is about solving for the basic variables and in the hand computation that we have seen we do it using the Gauss Jordan method and we also know that solving for equations is the same as inverting a matrix and we have effectively used the Gauss Jordan method to do that.

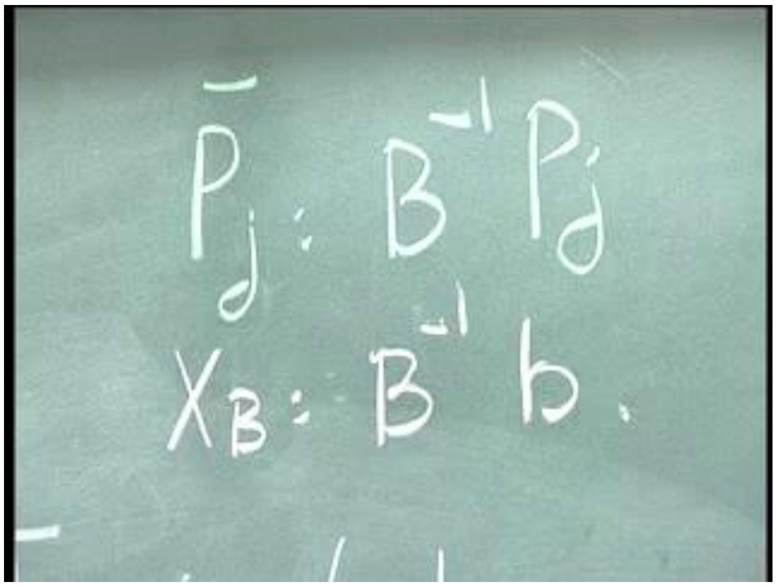
Now the next question is what is the computational effort that is required in these iterations?
Now if we see these iterations if we really follow the simplex what we actually require is if these two are the basic variables first and foremost we want the values of Cj – Zj corresponding to these basic variables. We really don't need to compute all these. All we need is to compute the Cj – Zj of the nonbasic variables which are X_2 and X_4 we just need these two values because we already know that Cj – Zj for the basic variables will be 0. So these two values we need to compute. If these two values are less than or equal to 0 then we have reached the optimal solution. So we just need to compute these values first and when we compute these two values we will know whether the present solution is optimal or some other thing is required or one more iteration is required.

Now, once we know that another iteration is required then you need to find out which of these variables enters and which of these variables leave. So in order to find the entering variable, let's assume that between these two this variable enters (refer Slide Time: 56:50) so this variable enters then we require this column and we require this column so that the leaving variable can be found out and the next one can go.

If it is optimal then we require only this so that we can find out the solution and also find out the value of the objective function. so in some sense what we require first are the Cj – Zj values we require this value (refer Slide Time: 57:24). Now here we have n – m values because there are n variables out of which m variables are basic variables which will have 0 so the remaining n – m variables will only have values which are nonzero and it is enough to find out this n – m values.

Once we know that there is an entering variable we need to find out this column which is another m values, we need to find out this column which is another m values and then we need to find out the leaving variable which is a simple division and then proceed. But in order to find out this value and this value we have already seen in the earlier course that this is this is given by the by the expression P bar j is B inverse P_j and it is also given as X_B = B inverse.

(Refer Slide Time; 00:58:30 min)

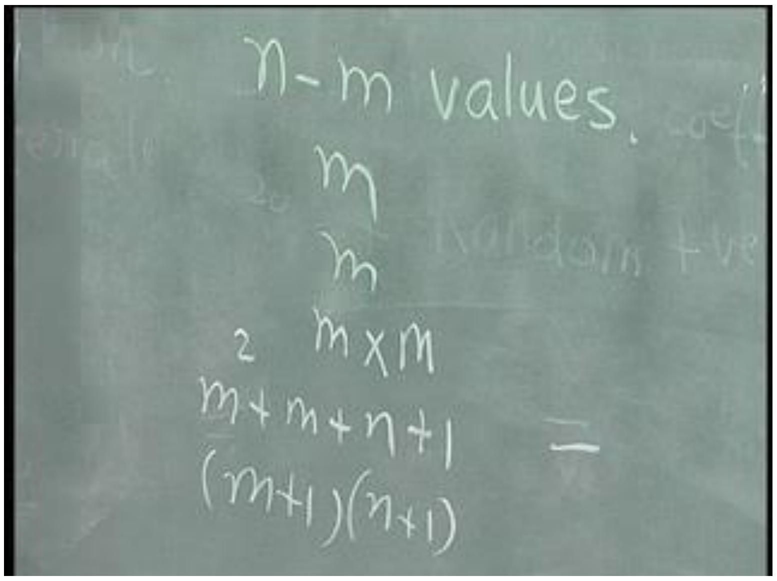

$$\bar{P}_j : B^{-1} P_j$$
$$X_B : B^{-1} b.$$

Now what are these Pj and B? Pj is the corresponding value in the first iteration. B is the corresponding right hand side values which are known. Now what is required further is this B inverse. And we have already seen in the earlier course that B inverse appears under the original identity matrix which means we also require this which also means we require another m into m. So every iteration actually requires m square + m + n computations + 1 if u need so that many computations are enough. But what we end up doing is we actually evaluate completely all these which is effectively m + 1 into n + 1.

So can we have an efficient algorithm which will evaluate fewer numbers than evaluating this entire rectangle which is m + 1 into n + 1. So that is the first thing if we wish to make simplex better in terms of the number of computations and in terms of the memory and storage particularly storage required for the simplex. We also realize that out of these computation of B inverse is the most intensive or intense computation while the rest of them are only matrix multiplications. So a large amount of effort in the simplex iteration actually goes in finding out the B inverse. So there is a basis matrix associated with every iteration and a large amount of computational effort goes in trying to find out the B inverse.

Now in this method we actually find out B inverse using the Gauss Jordan method. So the focus shifts on trying to find out the best way to invert a matrix if you really want to solve a simplex iteration quickly.

(Refer Slide Time: 1:02:00)


n - m values
m
m
2 m x m
m + m + n + 1 =
(m+1)(n+1)

So the speed of the simplex algorithm depends on the time it takes to invert a matrix. So the emphasis shifts from performing these iterations to saying how quickly can I invert this matrix. So there are two issues in making the simplex faster. One is to find out and efficient way of matrix inversion and the second is to make minimum computation that is required and not more than the minimum computation that is required. So we will look at both these aspects of how well we can invert this matrix and how do we develop a method by which minimum computation is required by taking a numerical example.