257

P. G .D. O. R. M.
SEM - II

ADVANCED LINEAR PROGRAMMING

Programme Co-ordinator : **Dr. Madhura Kulkarni**
Asst. Prof..-Cum. Asst. Director
IDOL, University of Mumbai

Course Co-ordinator : **Ms. Megha Bansal**
IDOL, University of Mumbai

Course Writer : **Prof. Radha Iyer**
ADMI, University of Mumbai

Reprint October 2018, P.G.D.O.R.M. SEM - II,
ADVANCED LINEAR PROGRAMMING

Published by : Incharge Director
Institute of Distance and Open Learning ,
University of Mumbai,
Vidyanagari, Mumbai - 400 098.

DTP Composed : Ashwini Arts
Gurukripa Chawl, M.C. Chagla Marg, Bamanwada,
Vile Parle (E), Mumbai - 400 099.

Printed by : P Square Solutions, Barari, Mathura

CONTENTS

❖❖❖❖

P.G.D.O.R.M.
SEM – II
ADVANCED LINNER PROGRAMMING
Syllabus

1) Dynamic Programming :

Introduction to Dynamic Programming, Dynamic Programming Approach, Formulation of Dynamic Programming, Optimal Sub-division Problem, System Reliability.

2) Integer Programming :

Introduction to Integer Programming, Types of Integer Programming, Gomory's Integer Cutting Plane Method, Gomory's Mixed Integer Cutting Plane Method, Branch and Bound Method

3) Goal Programming :

Introduction to Goal Programming, the Concept of Goal Programming, General Goal Programming Method, Modified Simplex Method of Goal Programming.

4) Parametric Programming :

Introduction to parametric Programming, Variation in the Objective Function Coefficient, Variability in the availability of Resources

5) Non Linear Programming Methods :

Introduction, General Non-Linear Programming Problem, Graphical Solution Method, Quadratic Programming.

6) Software Applications in OR :

Introduction to various software's used in OR, Understanding the solver add-in to solve OR problems, Understanding various other software's like SPSS, LINDO etc. for Optimisation.

References :

1) Quantitative Techniques in Management, N.D. Vohra, McGraw Hill.
2) Operations Research, Premkumar Gupta, D.S. Hira, S. Chand.
3) Operations Research, J.K. Sharma, MacMillan.
4) Business Statistics, Naval Bajpai, Pearson.
5) Business Mathematics, Zameeruddin Kazi, Vijay Khanna, S.K. Bhambri, Vikas Publication.
6) Business Statistics, J.K. Sharma, Pearson.

❖❖❖❖

1

DYNAMIC PROGRAMMING

Unit Structure

1.1 INTRODUCTION

In optimisation problems involving a large number of decision variables or the inequality constraints, it may not be possible to use the methods of calculus for obtaining a solution. Classical mathematics handles the problems in a way to find the optimal values for all the decision variables simultaneously which for large problems rapidly increases the computations that become uneconomical or difficult to handle even by the available computers. The obvious solution is to split up the original large problem into small sub-problems involving a few variables and that is precisely what the dynamic programming does.

Dynamic programming is a mathematical technique dealing with the optimisation of multistage decision problems. The technique was originated in 1952 by Richard Bellman and G.B. Dantzig, and wan initially referred to as the stochastic linear programming. Today dynamic programming has been developed as a mathematical technique to solve a wide range of decision problems and it forms an important part of every operation researcher's tool kit.

Though the originator of the technique, Richard Bellman, himself, has said, "we have coined the term 'dynamic programming' to emphasise that there are problems in which time plays an essential role", yet, in many dynamic programming problems time is not a relevant variable. For example, a decision regarding allocation of a fixed quantity of resources to a number of alternative uses constitutes one decision to be taken at one time, but the situation can be handled as a dynamic programming problem. As another instance, suppose a company has marked capital C to be spent on advertising its products through three different media i.e., of newspaper, radio and television. In each media the advertisement can appear a number of times per week. Each appearance has associated with it certain costs and returns. How many times the product should be advertised in each media so that the returns are maximum and the total cost is within the prescribed limit? In this situation time is not a variable, but the problem can be divided into stages and solved by dynamic programming.

1.2 CHARACTERISTICS OF DYNAMIC PROGRAMMING

The important features of dynamic programming which distinguish it from other quantitative techniques of decision-making can be summarized as follows:

1. It involves a multistage process of decision-making. The stages may be certain time intervals or certain sub-division of the problems, for which independent feasible decisions are possible.

2. In dynamic programming, 'state' is a description of the system (problem) which tells the necessary parameters of the system for the purpose of making decisions. It is not essential to know about the previous decisions and how the states arise. This enables us to consider decisions one at a time.

3. In dynamic programming the outcome of decisions depends upon a small number of variables; that is, at any stage only a few variables should define the problem. For example, in the production smoothening problem, all that one needs to know at any stage is the production capacity, cost of production in regular and overtime, storage costs and the time remaining to the last decision.

4. A stage decision does not alter the number of variables on which the outcome depends, but only changes the numerical value of these variables. For the production smoothening problem, the number of variables which describe the problem i.e., production

capacity, production costs, storage costs and time to the last decision, remain the same at all stages. No variable is added or dropped. The effect to decision at any stage will be to alter the used production capacity, storage cost, production cost and time remaining to the last decision.

5. Principle of Optimality: Dynamic Programming is based on Bellman's Principle of Optimality, which states, "An optimal policy (a sequence of decision) has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision". This principle implies that a wrong decision taken at one stage does not prevent from taking of optimum decisions for the remaining stages. For example, in a production scheduling problem, wrong decisions made during first and second months do not prevent taking correct decisions during third, fourth month etc.

6. Bellman's principle of optimality forms the basis of dynamic programming technique. With this principle in mind, recursive equations are developed to take optimal decision at each stage. A recursive equation expresses subsequent state conditions and it is based on the fact that a policy is 'optimal' if the decision made at each stage results in overall optimality over all the stages and not only for the current stage.

7. Dynamic Programming provides a systematic procedure wherein starting with the last stage of the problem and working backwards one makes an optimal decision for each stage of the problem. The information for the last stage is the information derived from the previous stages. It may be noted that D.P. problems can also be solved by working forward i.e., starting with the first stage and then working forward up to the last stage.

1.3 DYNAMIC PROGRAMMING APPROACH

Before discussing the solutions to numerical problems, it will be worthwhile to know a little more about some fundamental concepts of dynamic programming. The first concept is stage. As already discussed, the problem is broken down into sub problems and each sub problem is referred to as a stage. A stage signifies a portion of decision problem for which a separate decision can be made. At each stage there are a number of alternatives and the decision-making process involves the selection of one feasible alternative which may be called as stage decision. The stage decision may not be optimal for the considered stage, but contributes to make an overall optimal decision for the entire problem.

The other important concept is state. The variables which specify the condition of decision process and summarise the current 'status' of the system are called state variables. For example, in the capital budgeting problem, the capital is the state variable. The amount of capital allocated to the present stage and the preceding stages (or the capital remaining) defines the status of the problem. The number of state variables should be as small as possible. With the increase in number of state variables, increases the difficulty of problem solving.

The procedure adopted in the analysis of dynamic programming problems can be summarized as follows:

1. Define the problem variables, determine the objective function and specify the constraints.
2. Define the stages of the problem. Determine the state variables whose values constitute the state at each stage and the decision required at each stage. Specify the relationship by which the state at one stage can be expressed as a function of the state and decision at the next stage.
3. Develop the recursive relationship for the optimal return function which permits computation of the optimal policy at any stage. Specify the optimal return function at stage 1, since it is generally a bit different from the general optimal return function for the other stages.
4. Make a tabular representation to show the required values and calculations for each stage.

1.4 FORMULATION OF DYNAMIC PROGRAMMING PROBLEMS

Consider a situation where in a certain quantity 'R' of a resource (such as men, machines, money, materials etc.) is to be distributed among 'n' number of different activities. The return 'P' depends upon the activities and the quantities of resource allotted to them and the objective is to maximise the total return.

If $p_i (R_i)$ denotes the return form the *ith* activity with the resource R_i, then the total return may be expressed as

$$P(R_{1,} R_{2,} ......,R_n) = p_1(R_1) + p_2 (R_2) + ...+ p_n (R_n) ....(1)$$

The quantity of resource R is limited, which gives rise to the constraint

$$R = R_1 + R_2 + ...+ R_n, R_i \geq 0, i = 1, 2, ...,n. ....(2)$$

The problem is to maximise the total return given by equation (1) subject to constraint (2),

$$\text{If } f_n (R) = Max_{0\leq R1\leq R} [P(R_1, R_{2,} ....,R_n)] = Max_{0\leq R1\leq R} [p_1(R_1) + p_2 (R_2) + ...+ p_n (R_n)] .....(3)$$

then f_n (R) is the maximum return from the distribution of the resource R to the n activities. Let us now allocate the resource to the activities, one by one, starting from the last i.e. nth activity. An expression connecting f_n (R) and f_{n-1} (R) for arbitrary values of R and n may now be obtained with the help of principle of optimality. If R_n is the quantity of resources allocated to the nth activity such that $0 \leq R_n \leq R$, then regardless of the values of R_n, a quantity (R-R_n) denote the return from the (n-1) activities, then the total return from all the n activities will be

$$p_n (R_n) + f_{n-1} (R- R_{n).}$$

An optimal choice of R_n will maximise the above function and thus the fundamental dynamic programming model may be expressed as

$$f_n (R) = Max_{0 \leq Rn \leq R} [p_n (R_n) + f_{n-1} (R - R_n)], n = 2, 3, ...(4)$$

where f_1 (R), when n = 1 is obtained from equation (3) as

$$f_1 (R) = p_1 (R). \quad ....(5)$$

Equation (5) gives the return from the first activity when whole of the resource R is allotted to it. Once f_1 (R) is known, equation (4) provides a relation to evaluate f_2 (R), f_3 (R),.....This recursive process ultimately leads to the value of f_{n-1}(R) and finally f_n (R) at which the process stops.

EXAMPLE 1: (Employment Smoothening Problem)

A firm has divided its marketing area into three zones. The amount of sales depends upon the number of salesmen in each zone. The firm has been collecting the data regarding sales and salesmen in each area over a number of past years.

The information is summarised in table 1. For the next year firm has only 9salesmen and the problem is to allocate these *salesmen to three different zones so that the total sales are maximum.*

Table 1

Profits in thousands of rupees

No. of Salesmen	*Zone 1*	*Zone 2*	*Zone 3*
0	30	35	42
1	45	45	54
2	60	52	60
3	70	64	70
4	79	72	82
5	90	82	95
6	98	93	102
7	105	98	110
8	100	100	110
9	99	100	110

Solution:

In this problem the three zones represent the three stages and the number of salesmen represent the state variables.

Stage 1: We start with zone 1. The amount of sales corresponding to different number of salesmen allocated to zone 1 are given in table 1 and are reproduced in table 2.

Table 2

Zone 1

No. of salesmen:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Profit (000' of Rs.):	*30*	*45*	*60*	*70*	*79*	*90*	*98*	*105*	*100*	*90*

Stage 2: Now consider the first two zones, zone 1 and 2. Nine salesmen can be divided among two zones in 10 different ways: as 9 in zone 1 and 0 in zone 2, 8 in zone 1 and 1 in zone 2, 7 in zone 1 and 2 in zone 2, etc. Each combination will have associated with it certain returns. The returns for all number of salesmen (total) 9, 8, 7, ..., 0 are shown in table 3.

For a particular number of salesmen, the profits for all possible combinations can be read along the diagonal. Max. profits are marked by*.

Table 3

Zone 1 X_1		0	1	2	3	4	5	6	7	8	9
$F_1(x_1)$ Zone 2 $f_2(x_2)$ X_2		30	45	60	70	79	90	98	105	100	90
0	35	65*	80*	95*	105*	114	125*	133	140	135	125
1	45	75	90	105	115*	124	135*	143*	150	145	
2	52	82	97	112	122	131	142	150	157		
3	64	94	109	124	134	143	154*	162			
4	72	102	117	132	142	151	162				
5	82	112	127	142	152	161					
6	93	123	138	153	163*						
7	98	128	143	158							
8	100	130	145								
9	100	130									

Stage 3: Now consider the distribution of 9 salesmen in three zones 1, 2 and 3. The decision at this stage will result in allocating certain number of salesmen to zone 3 and the remaining to zone 2 and 1 combined; and then by following the backward process, they will be distributed to zones 2 and 2.

For total of 9 salesmen to be allocated to the three zones, the returns are shown in table 4 below:

Table 4

No. of salesmen:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Total Profit $f_2(x_2) + f_1(x_1)$	65	80	95	105	115	125	135	143	154	163
Salesmen in Zone 2 + Zone 1 $(x_2 + x_1)$	0+0	0+1	0+2	0+3 1+2	0+4	0+5	1+5	3+4 1+6	3+5	6+3
No. of salesmen in Zone 3:	9	8	7	6	5	4	3	2	1	0
Profit $f_3(x_3)$:	110	110	110	102	95	82	70	60	54	42
Total Profit $f_3(x_3) + f_2(x_2) + f_1(x_1)$	175	190	205	207	210*	207	205	203	208	205

From table 4, the maximum profit for 9 salesmen is Rs. 2, 10, 000 if 5 salesmen are allotted to zone 3 and from the remaining four, 1 is allotted to zone 2 and 3 to zone 1.

EXAMPLE 2 CAPITAL BUDGETING PROBLEM

A manufacturing company has three sections producing automobile parts, bicycle parts and sewing machine parts respectively. The management has allocated Rs. 20, 000 for expanding the production facilities. In the auto parts and bicycle parts sections, the production can be increased either by adding new machines or by replacing some old inefficient machines by automatic machines. The sewing machine parts section was started only a few years back and thus the additional amount can be invested only by adding new machines to the section. The cost of adding and replacing the machines, along with the associated *expected returns in the different sections is given in the table 1. Select a set of expansion plans which may yield the maximum return.*

Alternatives	Auto parts Section		Bicycle Parts Section		Sewing Machine Parts Section	
	Cost (Rs.)	Returns (Rs.)	Cost (Rs.)	Returns (Rs.)	Cost (Rs.)	Returns (Rs.)
1. No Expansion	0	0	0	0	0	0
2. Add New Machines	4, 000	8, 000	8, 000	12, 000	2, 000	8, 000
3. Replace Old m/cs	6, 000	10, 000	12, 000	18, 000	-	-

Solution

Here each section of the company is a stage. At each stage there are a number of alternatives for expansion. Capital represents the state variable. Let us consider the first stage - the auto parts section. There are three alternatives: no expansion, add new machines and replace old machines. The amount that may be allocated to stage 1 may vary from 0 to Rs. 20, 000; of course, it will be overspending if it is more than Rs. 6000. The returns of the various alternatives is given in table 2.

Table 2

Stage 1: Auto parts section

State X_1 (000' of Rs.)	Evaluation of alternatives (Values in thousands of Rs.) 1	2	3	Optimal Solution	
	Cost C_{11} = 0 Return	Cost C_{12} = 4 Return	Cost C_{13} = 6 Return	Optimal Return	Decision
0	0	-	-	0	1
2	0	-	-	0	1
4	0	8	-	8	2
6	0	8	10	10	3
8	0	8	10	10	3
10	0	8	10	10	3
12	0	8	10	10	3
14	0	8	10	10	3
16	0	8	10	10	3
18	0	8	10	10	3
20	0	8	10	10	3

When the capital allocated is zero or Rs. 2, 000, only first alternative (no expansion) is possible. Return is, of course, zero. When the amount allocated is Rs. 4, 000, alternatives 1 and 2 are possible with returns of Rs. 0 and Rs. 8000. So, we select alternative 2 and when the amount allocated is Rs. 6000, all the three alternatives are possible, giving returns of zero, Rs. 8, 000 and Rs. 10, 000 respectively. So, we select alternative 3 with return of Rs. 10, 000 and so on.

Stage 2: Let us now move to stage 2. Here, again, three alternatives are available. The computations are carried out in table 3.

Table 3

Stage 1: Bicycle Parts Section (+ Auto parts section)

State X_2 (000' of Rs.)	Evaluation of alternatives (Values in thousands of Rs.) 1 2 3			Optimal Solution	
	Cost C_{21} = 0 Return	Cost C_{22} = 8 Return	Cost C_{23} = 12 Return	Optimal Return	Decision
0	0 + 0 = 0	-	-	0	1
2	0 + 0 = 0	-	-	0	1
4	0 + 8 = 8	-	-	8	1
6	0 + 10 = 10	-	-	10	1
8	0 + 10 = 10	12 + 0 = 12	-	12	2
10	0 + 10 = 10	12 + 0 = 12	-	12	2
12	0 + 10 = 10	12 + 8 = 20	18 + 0 = 18	20	2
14	0 + 10 = 10	12 + 10 = 22	18 + 0 = 18	22	2
16	0 + 10 = 10	12 + 10 = 22	18 + 8 = 26	26	3
18	0 + 10 = 10	12 + 10 = 22	18 + 10 = 28	28	3
20	0 + 10 = 10	12 + 10 = 22	18 + 10 = 28	28	3

Here state x_2 represents the total amount allocated to the current stage (stage 2) and the preceding stage (stage 1). Similarly, the return also is the sum of the current stage and the preceding stage (Principle of optimality). Thus when x_2 □ Rs. 8000, only the first alternative (no expansion) is possible. But with x_2 = Rs. 8, 000, a return of Rs. 12, 000, three alternatives are possible with second alternative (add new machines). With x_2 = Rs. 12, 000, three alternatives are possible with the maximum return of Rs. 20, 000 from alternative 2. The optimal policy consists of a set of two decisions, namely adopt alternative 2 at second stage (Table 3) and again alternative 2 at the first stage (table 2).

Stage 3: The computations for stage 3 are given in table 4.

Table 4

Stage 1: Sewing Machine Parts Section (+ Bicycle Parts Section + Auto parts section)

State X_3 (000' of Rs.)	Evaluation of alternative 2 (Values in thousands of rupees)		Optimal Solution	
	Cost $C_{21} = 0$ Return	Cost $C_{32} = 2$ Return	Optimal Return	Decision
0	0 + 0 = 0	-	0	1
2	0 + 0 = 0	8 + 0 = 8	8	2
4	0 + 8 = 8	8 + 0 = 8	8	1,2
6	0 + 10 = 10	8 + 8 = 16	16	2
8	0 + 12 = 12	8 + 10 = 18	18	2
10	0 + 12 = 12	8 + 12 = 20	20	2
12	0 + 20 = 20	8 + 12 = 20	20	1,2
14	0 + 22 = 22	8 + 20 = 28	28	2
16	0 + 26 = 26	8 + 22 = 30	30	2
18	0 + 28 = 28	8 + 26 = 34	34	2
20	0 + 28 = 28	8 + 28 = 36	36	2

For X_3 = 20, 000, the optimal decision for stage 3 is alternative 2, which gives a total return of Rs. 36, 000. This involves a cost of Rs. 2,000 and leaves Rs. 18, 000 to be allotted fro stages 2 and 1 combined. From table 3, for allocation of Rs. 18, 000, alternative 3 is to be chosen which costs Rs. 12, 000. For remaining sum of Rs. 6, 000 from table 2, decision alternative 3 is to be selected. Thus the optimal policy of expanding production facilities is 3-3-2, which can be elaborated as replace old machines with automatics in auto parts section, replace old machines with automatics in bicycle parts section, and add machines to the sewing machines parts section. This policy gives the optimal return of Rs. 36, 000.

EXAMPLE 3

A manufacturer has entered into a contract for the supply of the following number of units of a product at the end of each month:

Month:	*Jan*	*March*	*August*	*October*	*November*	*December*
No. of Units:	*10*	*5*	*20*	*3*	*6*	*30*

The units manufactured during a month are available for supply at the end of the month or they may be kept in storage at a cost of Rs. 2 per unit per month. Each time the manufacture of a batch of units is undertaken, there is a set-up cost of Rs. 400. Determine the production schedule which will minimise the total cost.

Solution

Here the six months represent the 6 stages and number of units to be manufactured are the state variables. We shall start from the last month of December and move backwards.

Month of December

The best decision is to produce 30 units with a cost of Rs. 400 towards the set-up cost and there is no storage cost.

Month of November

There are two alternatives:

1. Produce (6 + 30) = 36 units to satisfy the demand of November and December.
Total Cost = Rs. (400 + 30 x 2 x 1) = Rs. 460.

2. Produce 6 units in Nov. + 30 units in Dec. involving 2 set-ups and no storage cost.
Total cost = Rs. (400 + 400) = Rs. 800.

□ The optimum decision is to produce 36 units in Nov. and no units in Dec.

Month of October

Various alternatives are:

1. Produce (3 + 6 + 30) = 39 units in Oct.
Total cost = Rs. (400 + 6 x 2 x 1 + 30 x 2 x 2) = Rs. 532.
2. Produce (3 + 6) = 9 units in Oct. and 30 units in Dec.
Total Cost = Rs. (400 x 2 + 30 x w x 1) = Rs. 812.
3. Produce 3 units in Oct. and 36 units in Nov.

Total Cost = Rs. (400 x 2 + 30 x 2 x 1) = Rs. 860.
Note that as per the decision made in Nov., producing 3 units in Oct., 6 in Nov., and 30 in Dec. is already ruled out as it involves higher cost.

Thus optimum decision is to produce 39 units in Oct. and nothing in Nov. and Dec.

Month of August
The various possible alternatives are:
1. Produce (20 + 3 + 6 + 30) = 59 units in August.
Total Cost = Rs. (400 + 3 x 2 x + 6 x 2 x 3 + 30 x 2 x 4) = Rs. 688.
2. Produce (20 + 3 + 6) = 29 units in August and 30 in Dec.
Total Cost = Rs. (400 x 2) + 3 x 2 x 2 + 6 x 2 x 3) = Rs. 848.
3. Produce (20 + 3) = 23 units in August and 36 in Nov.
Total Cost = Rs. (400 x 2) + 3 x 2 x 2 + 30 x 2 x 1) = Rs. 872.
4. Produce 20 units in August and 39 in Oct.
Total Cost = Rs. (400 x 2) + 6 x 2 x 1 + 30 x 2 x 2) = Rs. 932.

Thus optimum decision is to produce 59 units in August and none in the following months.

Month of March
The various possible alternatives are:
1. Produce (5 + 20 + 3 + 6 + 30) = 64 units in March.
Total Cost = Rs. (400 + 20 x 2 x 5 + 3 x 2 x 7 + 6 x 2 x 8 + 30 x 2 x 9) = Rs. 1,278.
2. Produce (20 + 3 + 6) = 29 units in August and 30 in Dec.
Total Cost = Rs. (400 x 2 + 20 x 2 x 5 + 3 x 2 x 7 + 6 x 2 x 8) = Rs. 1,318.
3. Produce (5 + 20 + 3) = 28 units in March and 36 in Nov.
Total Cost = Rs. (400 x 2 + 20 x 2 x 5 + 3 x 2 x 7 + 30 x 2 x 1) = Rs. 1, 102.
4. Produce (5 + 20) = 25 units in March and 39 in Oct.
Total Cost = Rs. (400 x 2 + 20 x 2 x 5 + 6x 2 x 1+ 30 x 2 x 2) = Rs. 1, 132.
5. Produce 5 units in March and 59 units in August.
Total Cost = Rs. (400 x 2 + 3 x 2 x 2 + 6 x 2 x 3 + 30 x 2 x 4) = Rs. 1, 088.
 The optimum decision is to produce 5 units in March and 59 units in August. The total cost involved is Rs. 1, 088.

Month of January

The various possible alternatives are:

1. Produce all 74 units in January.

Total Cost = Rs. (400 + 5 x 2 x 2 + 20 x 2 x 7 + 3 x 2 x9 + 6 x 2 x 10 + 30 x 2 x 11) = Rs. 1, 534.

2. Produce (10 + 5 + 20 + 3 + 6) = 44 units in January and 30 units in December.

Total Cost = Rs. (400 x 2 + 5 x 2 x 2+ 20 x 2 x 7 + 3 x 2 x 9 + 6 x 2 x 10) = Rs. 1, 274.

3. Produce(10 + 5 + 20) = 35 units in January and 39 units in Nov.

Total Cost = Rs. (400 x 2 + 5 x 2 x 2 + 20 x 2 x 7 + 3 x 2 x 9 + 30 x 2 x 1) = Rs. 1, 214.

4. Produce (10 + 5 + 20) = 35 units in January and 39 units in Oct.

Total Cost = Rs. (400 x 2 + 5 x 2 x 2 + 20 x 2 x 7 + 6 x 2 x 1 + 30 x 2 x 2) = Rs. 1, 232.

5. Produce (10 + 5) 15 units in January and 59 units in August.

Total Cost = (400 x 2 + 5 x 2 x 2 + 3 x 2 x 2 + 6 x 2 x 3 + 30 x 2 x 4)

= Rs. 1, 108.

6. Produce 10 units in January, 5 units in March and 59 units in August.

Total Cost = (400 x 3 + 3 x 2 x 2 + 6 x 2 x 3 + 30 x 2 x 4) = Rs. 1, 488.

Thus optimum decision is to produce 15 units in Jan. and 59 units in August.

Therefore, the best production schedule that will minimise total cost and satisfy the demand from January till December is to produce 15 units in January and 59 units in August.

1.5 OPTIMAL SUBDIVISION PROBLEM

This problem deals with the division of a given quantity into a given number of parts. Let Q be the quantity to be divided in n number of parts ($u_1, u_2, u_3 \ldots u_n$). Then problem can be expressed as maximise $\prod_{i=1}^{n} ui$ or maximise $u_{1,} u_{2,} u_3 \ldots\ldots u_n$

subject to $\sum_{i=1}^{n} ui$ = Q.

$u_i \geq 0$, i = 1,2, ..., n.

The problem can be handled by dynamic programming, by considering each part as a stage. The alternatives at each stage are infinite, since u_i is continuous and may assume any non-negative value, satisfying the constraints

$$\sum_{i=1}^{n} ui = Q.$$

The state of the system $x_{i,}$ at any stage i, represents the part of resource Q, allocated to stage 1 through i inclusive. The recursion formula is then given as

$f_1(x_1) = \max_{ui = xi} \{u_1\}$,

$fi(x_i) = \max_{0 \leq ui \leq xi} \{u_i f_{i-1}(x_i - u_i)\}$,

$= \max_{ui} \{ u_i ((x_i - u_i)\}$.

EXAMPLE 4

Determine the value of $u_{1,}$ $u_{2,}$ u_3 so as to maximise ($u_{1,}$ $u_{2,}$ u_3), subject to $u_1 + u_2 + u_3 = 10$ and $u_{1,} u_{2,} u_3 \geq 0$.

Solution:

In this example Q = 10, is to be divided into three parts *$u_{1,}$ u_2 and u_3* such that their product is maximum.

This D.P. problem can be regarded as a three-stage problem with sate variables $x_{1,}$ $x_{2,}$ x_3 and returns $f_1(x_1)$, $f_2(x_2)$ and $f_3(x_3)$ respectively.

At stage 3, $x_3 = u_1 + u_2 + u_3$

at stage 2, $x_2 = x_3 - u_3 = u_1 + u_2$

at stage 1, $x_1 = x_2 - u_2 = u_{1.}$

$f_1(x_1) = u_1 = x_2 - u_{2,}$

$f_2(x_2) = \max \{ u_2 (x_2 - u_2)\}$, $0 \leq u_2 \leq x_2$,

$= \max (u_2x_2 - u_2x_2^2)$, $0 \leq u_2 \leq x_{2.}$

Differentiating w.r.t $u_{2,}$ and equating the differential to zero,

$\frac{\partial f2\ (x2)}{\partial u2} = x_2 - 2u_2 = 0$ or $u_2 = \frac{x2}{2}$,

$f_2(x_2) = \frac{x2}{2}(x_2 - \frac{x2}{2}) = x_2^2/4.$

Now $f_3(x_3) = \max_{u3} \{ u_3 . x_2^2/4\} = \max_{u3}\{ u3 \frac{(x3 - u3)2}{4} \}$

Differentiating w.r.t u_3 and equating to zero,

$$\frac{\partial}{\partial u}[(u_3 x_3^2 + u_3^3 - 2 u_3^2 x_3)]/4 = 0$$
$$\text{or } x_3^2 + 3 u_3^2 - 4 u_3 x_3 = 0$$
$$\text{or } x_3^2 - 3 u_3 x_3 + 3 u_3^2 - u_3 x_3 = 0$$
$$\text{or } x_3 (x_3 - 3 u_3) - u_3(x_3 - 3 u_3) = 0$$
$$\text{or } (x_3 - u_3)(x_3 - 3 u_3) = 0.$$

□ Either $x_3 = u_3$ which is trivial since

$$x_3 = u_1 + u_2 + u_3;$$
$$\text{or } u_3 = \frac{x3}{3} = \frac{10}{3},$$
$$□\ x_2 = 10 - \frac{10}{3} = \frac{20}{3},$$
$$\text{and } u_2 = \frac{x2}{2} = \frac{10}{3} \text{ and hence } u_1 = \frac{10}{3},$$
$$□\ u_1 = u_2 = u_3 = \frac{10}{3}$$

and maximum product = $u_1 . u_2 . u_3 = \frac{1,000}{27}$.

SYSTEM RELIABILITY

EXAMPLE 5

An electronic device consists of four components, each of which must function for the system to function. The system reliability can be improved by installing parallel units in one or more of the components. The reliability (R) of a component with one, two or three parallel units and the corresponding cost (C) are given in Table 1. The maximum amount available for this device is 100. The problem is to determine the number of parallel units in each component.

TABLE 1

Number of units	*Components*			
	1 R C	*2 R C*	*3 R C*	*4 R C*
1	*0.70 10*	*0.50 20*	*0.70 10*	*0.60 20*
2	*0.80 20*	*0.70 40*	*0.90 30*	*0.70 30*
3	*0.90 30*	*0.80 50-*	*0.95 40*	*0.90 40*

Solution:

The reliability of a system is the product of the reliability of its components. If R_i u_i is the reliability of component having u_i units in parallel, then the reliability of the system comprising of n components in series is $\prod_{i=1}^{n} ui$ R_i $u_{i.}$ The problem then becomes

$$\text{maximise } R = \prod_{i=1}^{n} ui \ R_i \ u_{i,}$$

$$\text{subject to } \sum_{i=1}^{n} Ci.ui \leq C,$$

where $Ci.ui$ is the cost of components, when it has ui units in parallel, and C is the total capital available. The problem can be solved by considering the components as stages and the capital allocated as state of the system, x_i. The state x_i ($0\leq x_i \leq C$) is the capital allocated to stages 1 through i, inclusive. The reliability of the components of the return function at stage i may be expressed as f_i (x_i).

The recursive equation can be written as

$$f_1(x_1) = \max_{ui} \{ R_i.u_i\},\ 0 \leq Ci.ui \leq x_1;$$

$$f_i(x_i) = \max_{ui} \{ R_i.u_i \times f_{i-1}(x_i - Ci.ui) \},\ 0 \leq Ci.ui \leq x_{i.}$$

Since the return functions of different components are multiplied by each other, the procedure is called *multiplicative decomposition.*

In the given example, device will consist of at least one unit in each component.

$$\therefore C_{11} \leq x_1 \leq C - C_{21} - C_{31} - C_{41} \qquad \text{or } 10 \leq x_1 \leq 50,$$
$$C_{11} + C_{12} \leq x_2 \leq C - C_{31} - C_{41} \qquad \text{or } 30 \leq x_2 \leq 70,$$
$$C_{11} + C_{12} + C_{13} \leq x_3 \leq C - C_{41} \qquad \text{or } 40 \leq x_3 \leq 80,$$
$$C_{11} + C_{12} + C_{13} + C_{14} \leq x_4 \leq C \qquad \text{or } 60 \leq x_4 \leq 100.$$

The computations for different stages are given below in the tabular form.

Stage 1

	$F_1 (u_1/x_1) = R_i.u_i$			Optimal Solution

x_i	u_i =1
	R = 0.7,
	C = 10
		u_i =2
		R = 0.8,
		C = 20
			u_i =3
			R = 0.9,
			C = 30
				$f_1 (x_1)$
				u_i

10	.7	-	-	.7 1
20	.7	.8	-	.8 2
30	.7	.8	.9	.9 3
40	.7	.8	.9	.9 3
50	.7	.8	.9	.9 3

Stage 2

	$F2 (u_2/x_2) = R_2.u_2.f_1(x_2 - C_2.u_2)$			Optimal Solution

x_i	$u2$ =1
	R = 0.5,
	C = 20
		u_i =2
		R = 0.8,
		C = 20
			u_i =3
			R = 0.9,
			C = 30
				$F_2 (x_2)$
				u_2

30	.5 x .7	-	-	.35 1
40	.5 x .8	-	-	.40 1
50	.5 x .9	.7 x .7	-	.45 2
60	.5 x .9	.7 x .8	.8 x .7	.45 2,3
70	.5 x .9	.7 x .9	.8 x .8	.45 3

Stage 3

	$F_3(u_3/x_3) = R_3.u_3.f_2(x_3 - C_3.u_3)$			Optimal Solution

X_i	$u3$ =1
	R = 0.7,
	C = 10
		$u3$ =2
		R = 0.9,
		C = 30
			$u3$ =3
			R = 0.95, C = 40
				$F_3(x_3)$
				u_3

40	.7 x .35 = .245	-	-	.245 1
50	.7 x .40 = .280	-	-	.280 1
60	.7 x .49 = .343	.9 x .35 = .315	-	.343 1
70	.7 x .56 = .392	.9 x .40 = .360	.95 x .35 = .3325	.392 1
80	.7 x .64 = .448	.9 x .49 = .441	.95 x .40 = .380	.448 1

Stage 4

	$F_4(u_4/x_4) = R_4.u_4.f_3(x_4 - C_4.u_4)$			Optimal Solution
X_i	$u4$ =1 R = 0.6, C = 20	$u4$ =2 R = 0.7, C = 30	$u4$ =3 R = 0.9, C = 40	$F_4(x_4)$ u_4
60	.6 x .245 = .147	-	-	.147 1
70	.6 x .280 = .168	.7 x .245 = .1715	-	.168 2
80	.6 x .343 = .2058	.7 x .280 = .196	.9 x .245 = .2205	.2058 3
90	.6 x .392 = .2352	.7 x .343 = .2401	.9 x .280 = .252	.2352 1
100	.6 x .448 = .2688	.7 x .392 = .2744	.9 x .343 = .3087	.2688 3

Optimal value of $F_4(x_4) = 0.3087$ with $u_4 = 3$ and $x_4 = 100$, is obtained from $F_3(x_3) = 0.343$ which has $u3$ =1 and $F_2(x_2) = .49$, which is for $u2$ =2 and then $u1$ =1. Thus the optimal allocation is :3 units in parallel should be installed on component four, 1 unit on component three, 2 units on component two and 1 unit on component one.

1.5 APPLICATIONS OF DYNAMIC PROGRAMMING

Linear programming has found its applications in large-scale complex situations, dynamic programming has more applications in smaller-scale systems. Following are a few of the large number of fields in which dynamic programming has been successfully applied:

1. Production: In the production area, this technique has been employed for production scheduling and employment smoothening, in the face of widely fluctuating demand requirements.

2. Inventory Control: This technique has been used to determine the optimum inventory level and for formulating the inventory reordering rules, indicating when to replenish an item and by what amount.

3. Allocation of Resources: It has been employed for allocating the scarce resources to different alternative uses, such as allocating salesman to different sales zones and capital budgeting procedures.

4. Selection of an advertising media: (See example)

5. Spare Part Level Determination: To guarantee high efficiency utilisation of expensive equipment.

6. Equipment replacement policies: To determine at which stage equipment is to be replaced for optimal return from the facilities.

7. Scheduling methods for routine and major overhauls on complex machinery.

8. Systematic plan or search to discover the whereabouts of a valuable resource.

These are only a few of the wide range of situations to which dynamic programming has been successfully applied. Many real operating systems call for thousands of such decisions. The dynamic programming models make it possible to make all these decisions, of course with the help of computers. These decisions individually may not appear to be much of economic benefit, but in aggregate they exert a major influence on the economy of a firm.

1.6 EXERCISES

1. What is dynamic programming? Write step-by-step procedure to solve a general problem by D.P. approach.
2. What is dynamic programming and what sort of problems can be solved by it?
3. Discuss dynamic programming application to business and develop the recursive relation used in dynamic programming formulation.
4. What is the need of dynamic programming and how is it different from linear programming? Write some applications of dynamic programming.
5. Write short note on characteristics of dynamic programming.

SOLVE:

1. A food processing firm has compiled the following data for future monthly production requirements and production costs in regular and overtime periods.

Month	Quantity	Cost per unit	
		Regular (Rs.)	Overtime (Rs.)
September	4, 000	20	30
October	5, 200	25	35
November	5, 000	24	34
December	3, 700	26	36
January	4, 200	20	30
February	3, 000	20	30

The production capacity of the firm is 6, 000 units in regular time and 3, 000 units in overtime. The cost of carrying storage is Rs. 7.50 unit per month. If at the end of August, there are 3, 500 units in stock at a cost of Rs. 25 each, what is optimal production schedule and the total associated cost? Note that no inventory is required at the end of six months.

2. A drug manufacturing concern has ten medical representatives working in three sales areas. The profitability for each representative in three sales areas is as follows:

No. of representatives		0	1	2	3	4	5	6	7	8	9	10
Profitability (thousands of rupees)	Area 1	15	22	30	38	45	48	54	60	65	70	70
	Area 2	26	35	40	46	55	62	70	76	83	90	95
	Area 3	30	38	44	50	60	65	72	80	85	90	85

Determine the optimum allocation of medical representatives in order to maximise the profits. What will be the optimum allocation if the number of representatives available at present is only six?

3. A company has three media A, B and C available for advertising its product. The data collected over the past years about the relationship between the sales and frequency of advertisement in the different media is as follows:

Frequency/month			
	A	B	C
1	125	180	300
2	225	290	350
3	260	340	450
4	300	370	500

The cost of advertisement is Rs. 5, 000 in medium A, Rs. 10, 000 in medium B and Rs. 20, 000 in medium C. The total budget allocated for advertising the product is Rs. 40, 000. Determine the optimal combination of advertising media and frequency.

❖❖❖❖

2

INTEGER PROGRAMMING

Unit Structure

2.1 INTRODCTION

A linear programming in which some or all of the variables must take non-negative integer (discrete) values is commonly referred to as integer linear programming problem. When all the variables are constrained to be integers, it is called an all (pure) integer programming problem, and in case only some of the variables are restricted to have integer values, the problem is said t be a mixed integer programming problem. In some situations each variable can take on the values of either zero or one, as in 'do' and 'not to do' type decisions; such problems are referred to as zero-one programming problems.

Strictly speaking, if in an L.P problem we restrict the variables to be non-negative integers, the problem becomes non-linear. However, it is convenient to still call it an integer linear programming problem because after dropping integer restrictions on the variables, the objective function as well as the constraints remain linear in form.

One obvious and common approach to solve an integer linear programming problem is to ignore the integer restrictions on the variables and to solve the resulting L.P. problem by any of the techniques described earlier, and then to round off or truncate the fractional values of the optimal solution to the nearest integers. This method, however, gives satisfactory results only if the values of the variables are very large so that rounding off or truncating results in negligible change. For example, if the optimum value of the decision variable comes out to be 4549 $\frac{5}{8}$, it can be easily rounded off to 4549 or 4550 without much error. However, for smaller values, rounding or truncating may produce a solution totally different from the true optimal integer solution. Some of the constraints may be violated (Solution may become infeasible) since rounding off certain variables may require substantial changes in the values of the other variables to satisfy all the constraints. For example, consider the constraints

$$-x_1 + x_2 \leq 5\frac{1}{2}$$

$$5x_1 + 3x_2 \leq 10\frac{1}{2}$$

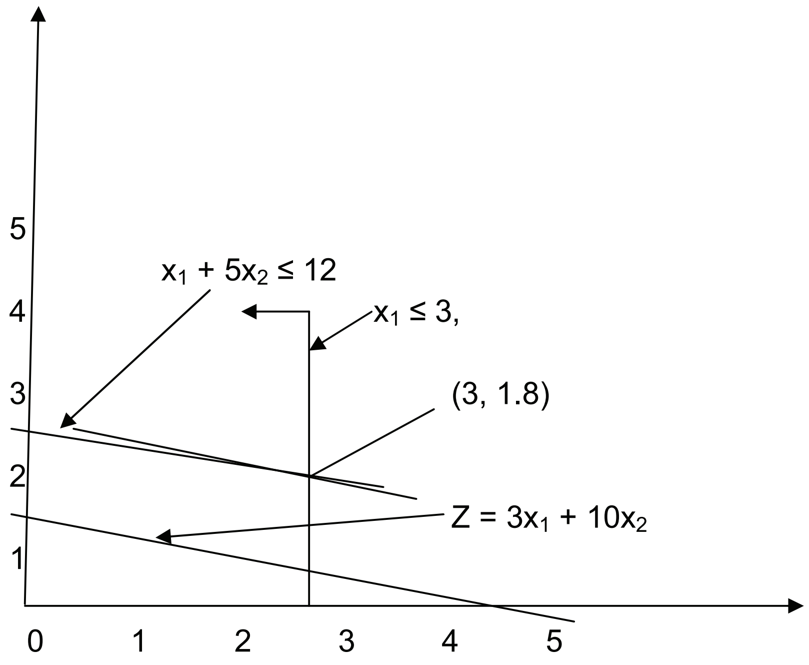
having non-integer optimal solution as $x_1 = 5\frac{1}{2}$ and $x_2 = 8$. Here rounding off x_1 to 2 or 3 or any other integer value violates one or the other constraint. Even if the rounded off solution satisfies the feasibility restriction, there is no guarantee that it is the true optimal integer solution. For instance, consider the problem

maximise $\quad Z = 3x_1 + 10x_2$
Subject to $\quad x_1 + 5x_2 \leq 12$
$\quad x_1 \leq 3,$
$\quad x_1, x_2$ non-negative integers.

The optimal non-integer solution of the above problem in figure 1 is $x_1 = 3$; $x_2 = 1.8$ and $Z_{max} = 27$.

If the non-integer variable is rounded off to 2, it violates the feasibility. Rounding off x_2 to 1 satisfies both the constraints but gives $Z_{max} = 19$, which is f less than the true optimal integer solution $x_1 = 2$; $x_2 = 2$ and $Z_{max} = 26$.

Figure 1



In addition, for large problems this method may be computationally expensive. For instance, if the optimal L.P. solution is $x_1 = 3.4$; $x_2 = 2.6$ and $Z_{max} = 8.6$, one has to try four alternatives, namely (3, 2), (3, 3), (4, 2) and (4, 3). The one which is feasible (satisfies all the constraints) and is closeset to the optimal value of 8.6 of the objective function will be the approximate integer solution. As the variables increase in number, the number of alternatives increases tremendously. For 3 variables, the number of alternatives is $2^3 = 8$, and for 11 variables it becomes $2^{11} = 2048$! And even after examining all such alternatives, the optimal integer solution to the problem is not guaranteed. All these difficulties justify the need for developing a systematic and efficient procedure for obtaining the exact optimal integer solution to such problems. A considerable number of algorithms have been developed for this purpose. Unfortunately, none possesses computational efficiency that is even remotely comparable to the simplex method (except on special types of problems), so they ordinarily are limited to relatively small problems having a few dozen variables. Despite decades of extensive research, computational experience with integer programming algorithms has been less than satisfactory. To date, there does not exist an I.P. computer code that can solve integer programming problems consistently. Therefore, this remains an active area of research and progress continues to be made in developing more efficient algorithms.

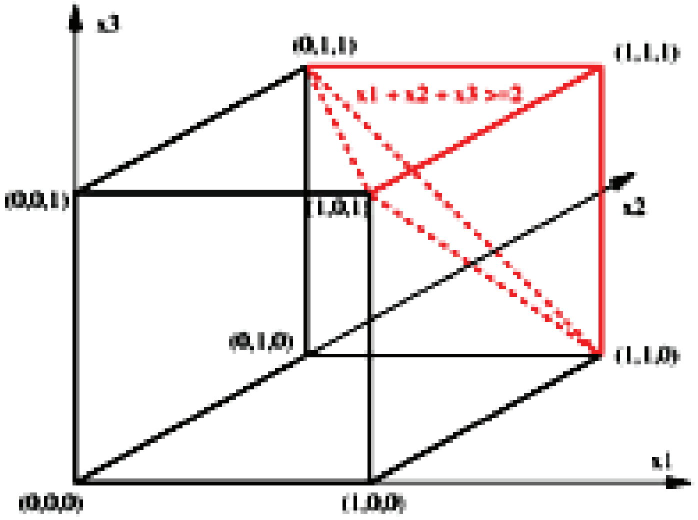
2.2 METHODS OF INTEGER PROGRAMMING

A systematic procedure for solving pure integer programming problems was first suggested by R. E. Gomory in 1958. He, later on extended the procedure to cover mixed integer programming problems. Named as cutting plane algorithm, the method consists in first solving the integer programming problem as ordinary continuous L.P. Problem and then introducing additional constraints one after the other to cut (eliminate) certain parts of the solution space until an integral solution is obtained.

Another method called the branch and bound algorithm originally developed by A.H. Land and A.G. Doig and later modified by R. J. Dakin's is more efficient and is more widely used for solving all integer and mixed integer programming problems. In this method, too, the problem is solved as ordinary continuous L. P. Problem and then the solution space is systematically 'partitioned' into subproblems by deleting parts that contain no feasible integer solutions.

The third algorithm, called additive algorithm due to E. Balas is an efficient and interesting algorithm for solving pure zero-one linear programming problems. The generalised penalty function method has also been developed to solve all integer and mixed integer programming problems.

2.3 THE CONCEPT OF CUTTING PLANE METHOD - ALGORITHMS



The intersection of the unit cube with the cutting plane .
In the context of the Traveling salesman problem on three nodes,

this (rather weak) inequality states that every tour must have at least two edges.

In mathematical optimization, the **cutting-plane method** is an umbrella term for optimization methods which iteratively refine a feasible set or objective function by means of linear inequalities, termed *cuts*. Such procedures are commonly used to find integer solutions to mixed integer linear programming (MILP) problems, as well as to solve general, not necessarily differentiable convex optimization problems. The use of cutting planes to solve MILP was introduced by Ralph E. Gomory.

Cutting plane methods for MILP work by solving a non-integer linear program, the linear relaxation of the given integer program. The theory of Linear Programming dictates that under mild assumptions (if the linear program has an optimal solution, and if the feasible region does not contain a line), one can always find an extreme point or a corner point that is optimal. The obtained optimum is tested for being an integer solution. If it is not, there is guaranteed to exist a linear inequality that *separates* the optimum from the convex hull of the true feasible set. Finding such an inequality is the *separation problem*, and such an inequality is a *cut*. A cut can be added to the relaxed linear program. Then, the current non-integer solution is no longer feasible to the relaxation. This process is repeated until an optimal integer solution is found.

Cutting-plane methods for general convex continuous optimization and variants are known under various names: Kelley's method, Kelley-Cheney-Goldstein method, and bundle methods. They are popularly used for non-differentiable convex minimization, where a convex objective function and its subgradient can be evaluated efficiently but usual gradient methods for differentiable optimization can not be used. This situation is most typical for the concave maximization of Lagrangian dual functions. Another common situation is the application of the Dantzig-Wolfe decomposition to a structured optimization problem in which formulations with an exponential number of variables are obtained. Generating these variables on demand by means of delayed column generation is identical to performing a cutting plane on the respective dual problem.

2.4 GOMORY CUTTING PLANE METHOD EXAMPLES: INTEGER PROGRAMMING

Cutting planes were proposed by Ralph Gomory in the 1950s as a method for solving integer programming and mixed-integer programming problems. However most experts, including Gomory himself, considered them to be impractical due to numerical instability, as well as ineffective because many rounds of cuts were needed to make progress towards the solution. Things turned around when in the mid-1990s Gérard Cornuéjols and co-workers showed them to be very effective in combination with branch-and-bound (called branch-and-cut) and ways to overcome numerical instabilities. Nowadays, all commercial MILP solvers use Gomory cuts in one way or another. Gomory cuts are very efficiently generated from a simplex tableau, whereas many other types of cuts are either expensive or even NP-hard to separate. Among other general cuts for MILP, most notably lift-and-project dominates Gomory cuts.

Let an integer programming problem be formulated (in Standard Form) as:

$$\text{Maximise } C^T{}_x$$
$$\text{Subject to } Ax = b,$$
$$x \geq 0,\ x_i \text{ all integers.}$$

The method proceeds by first dropping the requirement that the x_i be integers and solving the associated linear programming problem to obtain a basic feasible solution. Geometrically, this solution will be a vertex of the convex polytope consisting of all feasible points. If this vertex is not an integer point then the method finds a hyperplane with the vertex on one side and all feasible integer points on the other. This is then added as an additional linear constraint to exclude the vertex found, creating a modified linear program. The new program is then solved and the process is repeated until an integer solution is found.

Using the simplex method to solve a linear program produces a set of equations of the form

$$x_i + \sum \bar{a}_{ij}.x_j = \overline{b_i}$$

where x_i is basic variable and the x_j's are the non-basic variables. Rewrite this equation so that the integer parts are on the left side and the fractional parts are on the right side:

$$x_i + \left[\sum [\bar{a}_{ij}] x_j\right] - [\overline{b_i}] = \overline{b_i} - [\overline{b_i}] - \sum (\bar{a}_{ij} - [\bar{a}_{ij}])\ x_j$$

For any integer point in the feasible region the right side of this equation is less than 1 and the left side is an integer, therefore the common value must be less than or equal to 0. So the inequality

$$\overline{b_i} - [\overline{b_i}] - \sum (\bar{a}_{ij} - [\bar{a}_{ij}])\ x_j \leq 0$$

must hold for any integer point in the feasible region. Furthermore, non-basic variables are equal to 0s in any basic solution and if x_i is not an integer for the basic solution *x*,

$$\overline{b_i} - [\overline{b_i}] - \sum (\bar{a}_{ij} - [\bar{a}_{ij}])\ x_j = \overline{b_i} - [\overline{b_i}] > 0$$

So the inequality above excludes the basic feasible solution and thus is a cut with the desired properties. Introducing a new slack variable x_k for this inequality, a new constraint is added to the linear program, namely

$x_k + \Sigma \left[(\bar{a}ij - [\bar{a}ij] \right]) xj = [\overline{bi}] - \bar{b}i,$ $x_k \geq 0$, x_k is an integer.

In the previous section, we used **Gomory cutting plane method** to solve an Integer programming problem. In this section, we provide another example to enhance your knowledge. Let's concentrate on the following example:

EXAMPLE 1: GOMORY CUTTING PLANE METHOD

Maximize $z = x_1 + 4x_2$

subject to
$2x_1 + 4x_2 \leq 7$
$5x_1 + 3x_2 \leq 15$
x_1, x_2 are integers ≥ 0

Solution.

First, solve the above problem by applying the simplex method (try it yourself).The final simplex table is presented below.

Final Simplex Table

	c_j	**1**	**4**	**0**	**0**	
c_B	**Basic variables B**	**x_1**	**x_2**	**x_3**	**x_4**	**Solution values b (=X_B)**
4	x_2	1/2	1	1/4	0	7/4 (1 + 3/4)
0	x_4	7/2	0	-3/4	1	39/4 (9 + 3/4)
$z_j – c_j$		1	0	1	0	

Taking first row as the source row, the corresponding equation is
$1/2x_1 + 1x_2 + 1/4x_3 + 0x_4 = 1 + 3/4$
$1/2x_1 + (1 + 0)x_2 + (1 - 3/4)x_3 = 1 + 3/4$

Gomory's constraint
$- (1/2x_1 - 3/4x_3) \leq -3/4$
$-1/2x_1 + 3/4x_3 + x_5 = -3/4$

Table

	c_j	1	4	0	0	0	
c_B	**Basic variables B**	x_1	x_2	x_3	x_4	x_5	**Solution values b ($=X_B$)**
4	x_2	1/2	1	1/4	0	0	7/4
0	x_4	7/2	0	-3/4	1	0	39/4
0	x_5	-1/2	0	3/4	0	1	- 3/4
$z_j–c_j$		1	0	1	0	0	

Table

	c_j	1	4	0	0	0	
c_B	**Basic variables B**	x_1	x_2	x_3	x_4	x_5	**Solution values b ($=X_B$)**
4	x_2	0	1	1	0	1	1
0	x_4	0	0	9/2	1	7	9/2 (4+1/2)
1	x_1	1	0	-3/2	0	-2	3/2 (1+1/2)
$z_j–c_j$		0	0	5/2	0	2	

Taking second row as the source row, the corresponding equation is:
$0x_1 + 0x_2 + 9/2x_3 + 1x_4 + 7x_5 = 4 + 1/2$
or $(4 + 1/2)x_3 + (1 + 0)x_4 + (7 + 0)x_5 = 4 + 1/2$

Gomory's constraint
$-1/2x_3 \leq -1/2$
$-1/2x_3 + x_6 = -1/2$

Table

	c_j	1	4	0	0	0	0	
c_B	**Basic variables B**	x_1	x_2	x_3	x_4	x_5	x_6	**Solution values b (=X_B)**
4	x_2	0	1	1	0	1	0	1
0	x_4	0	0	9/2	1	7	0	9/2
1	x_1	1	0	-3/2	0	-2	0	3/2
0	x_6	0	0	-1/2	0	0	1	-1/2
$z_j–c_j$		0	0	5/2	0	2	0	

In the above table, there is a negative value under X_B column; therefore, apply the dual simplex method.

Final Table: Gomory Cutting Plane Method

	c_j	1	4	0	0	0	0	
c_B	**Basic variables B**	x_1	x_2	x_3	x_4	x_5	x_6	**Solution values b (=X_B)**
4	x_2	0	1	0	0	1	2	0
0	x_4	0	0	0	1	7	9	0
1	x_1	1	0	0	0	-2	-3	3
0	x_3	0	0	1	0	0	-2	1
$z_j–c_j$		0	0	0	0	2	5	

The **optimal solution** is
$x_1 = 3, x_2 = 0$
$z = 3 + 4 X 0 = 3$

2.5 THE MIXED ALGORITHM

In mixed integer problems only some of the variables are constrained to be integers. As in the case of pure integers, the problem is first solved as a continuous linear programming problem and then secondary cuts corresponding to the integer variables are added one by one. The value of the objective function in the optimum solution of the mixed integer programming problem is always superior to or at least equal to that of all integer problem, and is always inferior to or equal to that of the continuous L.P. Problem.

EXAMPLE 1:

Maximise $Z = 4x_1 + 6x_2 + 2x_3.$

subject to $4x_1 - 4x_2 \leq 5,$

$-x_1 + 6x_2 \leq 5,$

$-x_1 + x_2 + x_3 \leq 5;\ x_1, x_2, x_3 \geq 0;\ x_1, x_3$ integer.

Solution:

The optimal solution to this problem obtained by ignoring the integrality condition is given in the following simplex table:

Table 1

C_B	Basis	x_1	x_2	x_3	S_1	S_2	S_3	b
4	x_1	1	0	0	3/10	1/5	0	5/2
6	x_2	0	1	0	1/20	1/5	0	5/4
2	x_3	0	0	1	1/4	0	1	25/4

Optimal feasible non-integer solution

Variables x_1 and x_2 are constrained to be integers. To construct Gomory's constraint, select x_1-row which has the larger fractional part, 1/2. This row can be written as

$$(1 + 0)\, x_1 + (0 + 3/10)\, S_1 + (0 + 1/5)\, S_2 = 2 + 1/2$$

The Gomory's constraint to be added is

$$S' = 3/10\, S_1 + 1/5\, S_2 - 1/2$$

$$\text{or } -3/10\, S_1 - 1/5\, S_2 + S' = -1/2.$$

Adding this constraint to table 1, we get

Table 2

C_B	Basis	x_1	x_2	x_3	S_1	S_2	S_3	S'	**b**
4	x_1	1	0	0	3/10	1/5	0	0	5/2
6	x_2	0	1	0	1/20	1/5	0	0	5/4
2	x_3	0	0	1	1/4	0	1	0	25/4
0	S'	0	0	0	(-3/10)	-1/5	0	1	- 1/2 ←
-------	--------	-----	-----	-----	---------	------	-----	-----	-------
Z_j		4	6	2	2	2	2	0	
$\overline{c_j}$		0	0	0	-2	-2	-2	0	
ratio		-	-	-	20/3	10	-	-	

Replace S' by S_1.

Table 3

C_B	Basis	x_1	x_2	x_3	S_1	S_2	S_3	S'	b
4	x_1	1	0	0	0	0	0	1	2
6	x_2	0	1	0	0	1/6	0	1/6	7/6
2	x_3	0	0	1	0	-1/6	1	5/6	35/6
0	S_1	0	0	0	1	2/3	0	-10/3	5/3

Taking x_3-row as the source row, we have

$$(1 + 0)\, x_3 + (-1 + 5/6)\, S_2 + (1 + 0)\, S_3 + (0 + 5/6)\, S' = 5 + 5/6.$$

□The Gomory's constraint is

$$S'' = \frac{5}{6} S_2 + \frac{5}{6} S' - \frac{5}{6} \text{ or } -\frac{5}{6} S_2 - \frac{5}{6} S' + S'' = -\frac{5}{6}.$$

This constraint is now added to table 3 and we get

Table 4

C_B	Basis	x_1	x_2	x_3	S_1	S_2	S_3	S'	**S"**	**b**
4	x_1	1	0	0	0	0	0	1	0	2
6	x_2	0	1	0	0	1/6	0	1/6	0	7/6
2	x_3	0	0	1	0	-1/6	1	5/6	0	35/6
0	S'	0	0	0	1	(2/3)	0	-10/3	0	5/3
0	S"	0	0	0	0	(-5/6)	0	-5/6	1	-5/6 ←
-------	-------	----	-----	----	-----	-------	---	------		-------
Z_j		4	6	2	0	2/3	2	20/3	0	
$\overline{cj}$		0	0	0	0	-2/3	-2	-20/3	0	
ratio		-	-	-	-	4/5	-	8	-	
						↑				

Replace S" by S_2

Table 5

C_B	Basis	x_1	x_2	x_3	S_1	S_2	S_3	S'	S"	b
4	x_1	1	0	0	0	0	0	1	0	2
6	x_2	0	1	0	0	0	0	0	1/5	1
2	x_3	0	0	1	0	0	1	1	-1/5	6
0	S_1	0	0	0	1	0	0	-4	4/5	1
0	S_2	0	0	0	0	1	0	1	-6/5	1

Optimal Integer Solution

Table 5 gives optimal integer solution.

$x_1 = 2$; $x_2 = 1$; $x_3 = 6$; $Z_{max} = 26$.

2.6 BRANCH & BOUND (B&B) ALGORITHM

Branch and bound algorithm is the most widely used method for solving pure as well as mixed integer problems in practice. Most commercial computer codes use this method for solving I.P. Problems.

In this method also, the problem is first solved as a continuous I.P. problem ignoring the integrality condition. If in the optimal solution some variable, say x_1 is not an integer, then

$$x_j^* \square x_j \square x_j^* + 1.$$

where x_j^* and $x_j^* + 1$ are consecutive non-negative integers.

It follows that any feasible integer value of x_j must satisfy one of the two conditions, namely

$$x_j \leq x_j^* \text{ or } x_j \geq x_j^* + 1$$

since variable has no integer value between x_j^* and $x^*_{j+1.}$

These two conditions are mutually exclusive and when applied separately to the continuous L.P. problem, form two different sub problems. Thus the original problem is 'branched' or 'partitioned' into two sub problems. Geometrically it means that the branching process eliminates that portion of the feasible region that contains no feasible integer solution.

For instance, let the continuous optimal solution to a problem be

$$x_1 = 5\frac{1}{2},\ x_2 = 0,\ z_{max} = 55.$$

$$\text{Now, } x_1 = 5\frac{1}{2} \text{ gives } 5 \square x_1 \square 6.$$

For an integer valued solution,

$$\text{either } x_1 \leq 5 \text{ or } x_1 \geq 6$$

and we search for optimum value of Z either in the first region ($x_1 \leq 5$) or in the second region ($x_1 \geq 6$).

This branching process yields two sub problems, one by adding the constraint $x_j \leq x_j^*$ and the other by adding the constraint $x_j \geq x_j^* + 1$to the original set of constraints. Each of these sub problems is then solved separately as a linear program, using the same objective function of the original problem. If any sub problem yields an optimal integer solution, it is not further branched. However, it yields a non-integer solution, it is further branched into two sub problems. This branching process is continued, until each problem terminates with either integer valued optimal solution or there is evidence that it cannot yield a better one. Whenever a better integer solution is found for any sub problem, it replaces the one previously found. The integer valued solution, among all the sub problems that gives the most optimum value of the objective function is selected as the optimum solution.

Main drawback of this algorithm is that it requires the optimum solution of each sub problem and in large problems it could be very time-consuming. However, the computational efficiency of this algorithm is increased by applying the concept of "bounding". According to this concept, whenever the continuous optimum solution of a sub problem yields a value of the objective function lower than that of the best available integer solution (maximisation case), it is useless to explore the problem any further. This sub problem is said to be fathomed and is dropped from further consideration. Thus once a feasible integer solution is obtained, its associated objective function can be used as a lower bound (maximisation case) to delete inferior sub problems. Hence efficiency of a branch and bound algorithm depends upon how soon the successive sub problem are fathomed.

If the objective function is to be minimised, the procedure remains the same except that upper bounds are used. Thus the value of the first integral solution becomes an upper bound for the problem and the programmes are eliminated when their objective function values are greater than the current upper bound.

This algorithm can be extended directly to the mixed integer problems (in which only some of the variables are integer). If a variable is continuous, we simply never select it as a branching variable. A sub problem provides a new bound on the objective value if the values of all the discrete variables are integer and the objective value is better than the existing bound.

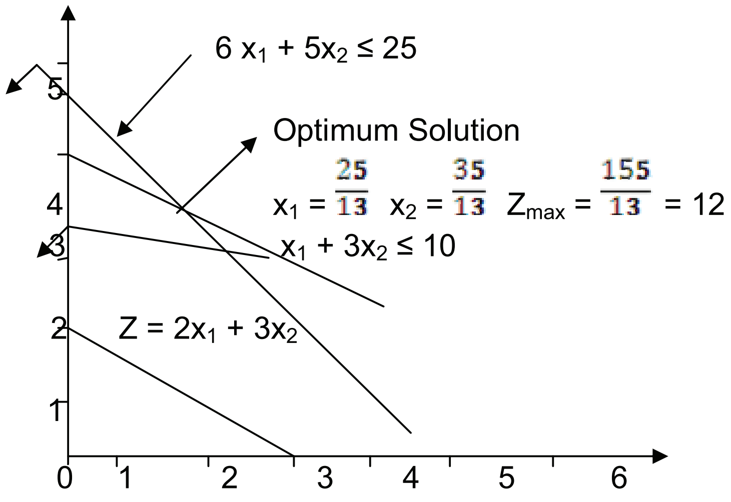
EXAMPLE 1

$$\begin{gathered}\text{Minimise} \quad Z = 2x_1 + 3x_2 \\ \text{Subject to } 6x_1 + 5x_2 \le 25 \\ x_1 + 3x_2 \le 10, \\ x_1, x_2 \text{ non-negative integers.}\end{gathered}$$

SOLUTION

The given problem is first solved as a continuous linear programming problem ignoring the integrality condition. Any of the methods, graphical or simplex could be used, however the graphical method will be used here. The non-integer optimal solution given in figure 1 is

Figure 1



$x_1 = \frac{25}{13}$ $x_2 = \frac{35}{13}$ $Z_{max} = \frac{155}{13} = 12$

This is the starting solution and is represented on the tree diagram given in figure 6. It can be observed that addition of integer restrictions can only make L. P. solution worse. Hence upper bound on the value of Z for the integer problem is 12. Since the solution is non-integer with both x_1 and x_2 having fractional values, any variable may be arbitrarily selected for branching. If x_2 is selected, then $x_2 = \frac{35}{13}$ gives 2 □ x_2 □ 3.

Thus, we add a new constraint either $x_2 \leq 2$ or $x_2 \leq 3$ to the original L.P problem, yielding two sub problems:

Sub Problem 1	*Sub Problem 1*
$Z = 2x_1 + 3x_2$	$Z = 2x_1 + 3x_2$
Subject to	Subject to
6 $x_1 + 5x_2 \leq 25$	6 $x_1 + 5x_2 \leq 25$
$x_1 + 3x_2 \leq 10$	$x_1 + 3x_2 \leq 10$
$x_2 \leq 2$	$x_2 \leq 3$
x_1, x_2 Non-negative integers	x_1, x_2 Non-negative integers

these sub problems are again solved by ignoring the internality condition. The solution of sub problem 1 given in figure 2 is
$x_1 = 2.5$, $x_2 = 2$; $Z_{max} = 11$

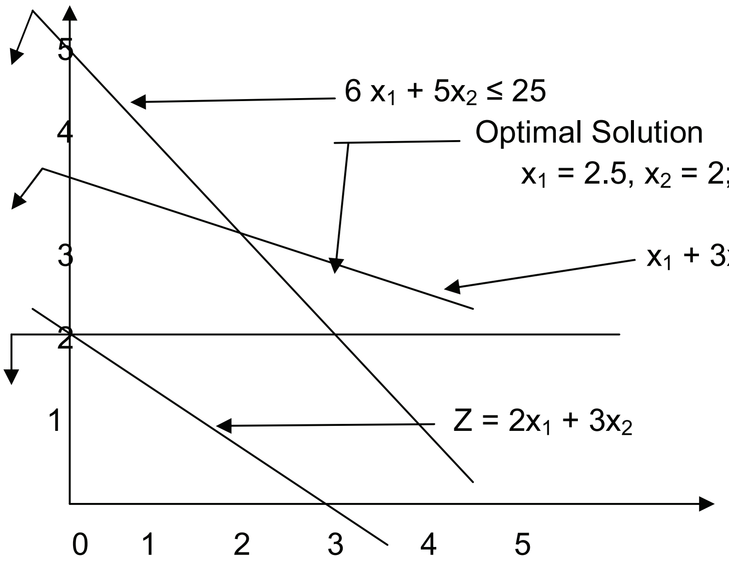


Figure 2

This is a feasible integer solution to the given problem. Since variables x_1 and x_2 are integers, there is no need to branch sub problem 2 further. Note that $Z_{max} = 11$ is a lower bound on the maximum value of Z for the future solutions. Thus $Z_L = 11$.

Similarly, the optimal solution obtained graphically in figure 3 for sub problem 2 is

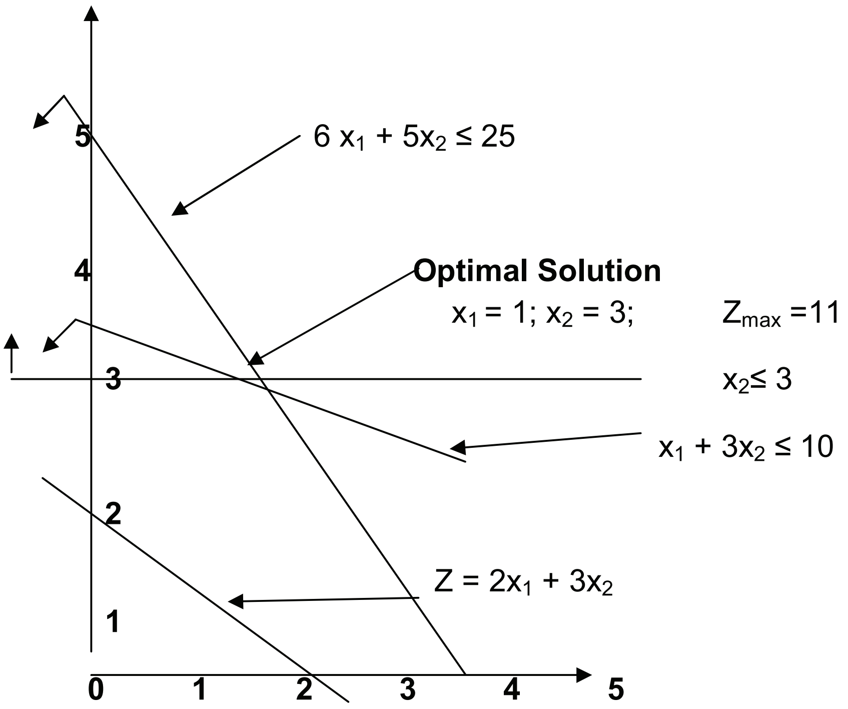


figure 3

Sub problem 1 is now the only candidate for branching. The optimum continuous solution to this problem give Z_{max} =11, which is not inferior to the lower bound. Therefore, it can be branched into further sub problems. Since x_1 is the only fractional valued variable, this will act as the branching variable and the new sub problems will have one of the following additional constraints:

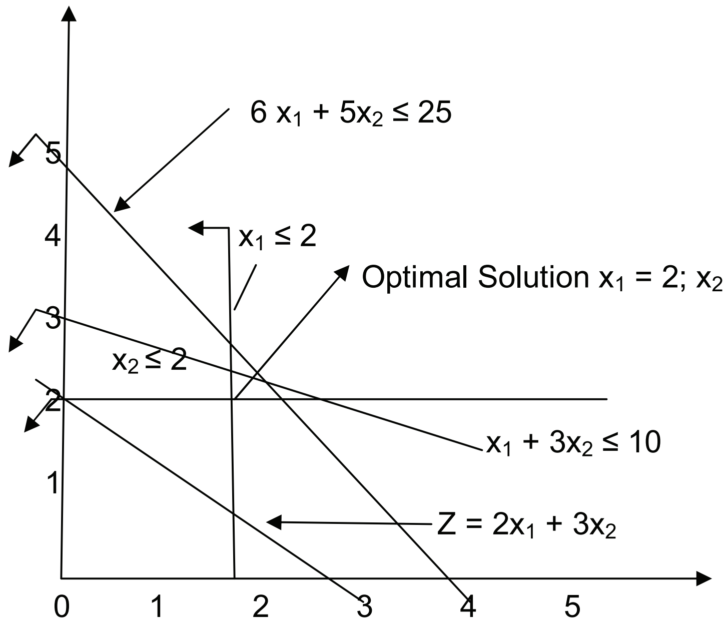
$x_1 \leq [2.5]$ or $x_1 \leq 2$,

$x_1 \geq [2.5]$ or $x_1 \geq 3$

Thus the sub problems emanating from it are

Sub problem 3	Sub problem 4
Maximise $Z = 2x_1 + 3x_2$	Maximise $Z = 2x_1 + 3x_2$
Subject to	Subject to
$6x_1 + 5x_2 \leq 25$	$6x_1 + 5x_2 \leq 25$
$x_1 + 3x_2 \leq 10$	$x_1 + 3x_2 \leq 10$
$x_2 \leq 2$,	$x_2 \leq 2$,
$x_1 \leq 2$,	$x_1 \leq 3$,
x_1, x_2 non-negative integers	x_1, x_2 non-negative integers

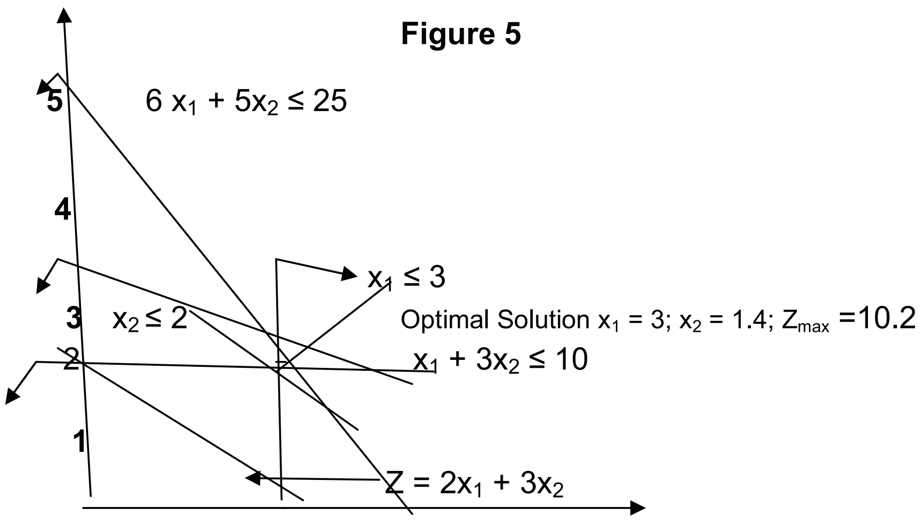
The optimal solution to sub problem 3, as shown in figure 4 is $x_1 = 2$; $x_2 = 2$; Z_{max} =10.



This solution is integer feasible but is inferior to the best available solution already obtained. Hence, the value of lower bound Z_L = 11 remains unchanged and sub problem 3 is also fathomed.

The optimal solution to sub problem 4, as shown in figure 5 is:

Figure 5



$x_1 = 3; x_2 = 1.4; Z_{max} = 10.2$

Since the solution is non-integer, sub problem 4 can be further branched with x_2 as the branching variable. But the value of its objective function ($Z_{max} = 10.2$) is inferior to the lower bound and hence this does not promise a solution better than the one already obtained. Therefore, this sub problem is also fathomed. Now there is no sub problem which can be further branched and the best available solution corresponding to sub problem 2 is the integer optimal solution of the problem.

i.e. $x_1 = 1; x_2 = 3; \quad Z_{max} = 11$

Figure 6

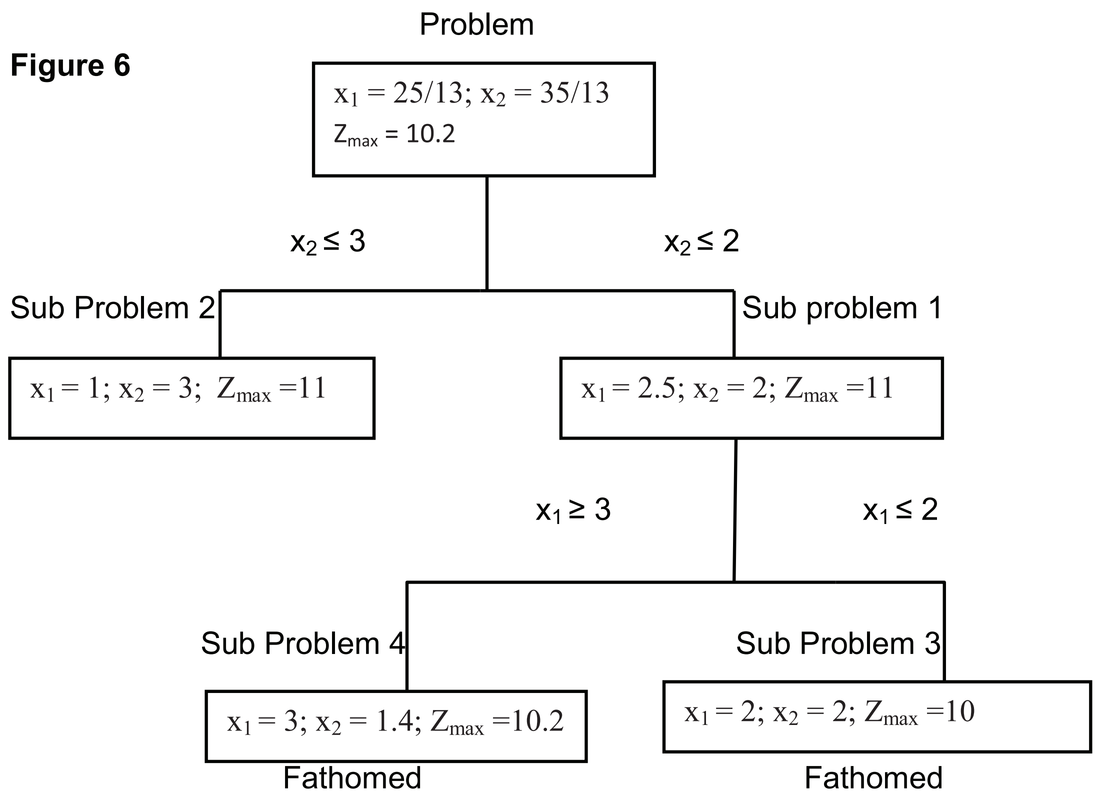


Figure 6 summarises the generated sub problems in the form of a tree.

The above example was solved using graphical method. We present below a mixed-integer problem and solve it by the simplex method.

EXAMPLE 2

Solve the following mixed integer problem by the branch and bound technique:

Maximise $z = X_1 + X_2$

Subject to $2x_1 + 5x_2 \leq 16$,

$6x_1 + 5x_2 \leq 30$,

$x_2 \geq 0$,

$x_1 \geq 0$ and integer

Solution

The continuous optimal solution of the problem has been obtained in table 1 as

$x_1 = 7/2$ and $x_2 = 9/5$, giving $Z_{max} = 53/10 = 5.3$

Table 1

	C_j	1	1	0	0		
C_B	**Basis**	x_1	X_2	s_1	S_2	**B_i**	**0**
0	S_1	**2**	**5**	**1**	**0**	**16**	**8**
0	S_2	**(6)**	**5**	**0**	**1**	**30**	**5** ←
	Z_j	**0**	**0**	**0**	**0**		
	$\overline{cj} = C_j - Z_j$	**1**	**1**	**0**	**0**		
		↑					
0	S_1	**0**	**(10/3)**	**1**	**-1/3**	**6**	**9/5**
1	x_1	**1**	**5/6**	**0**	**1/6**	**5**	**6** ←
	Z_j	**1**	**5/6**	**0**	**1/6**		
	$\overline{cj} = C_j - Z_j$	**0**	**1/6**	**0**	**-1/6**		
1	x_2	**0**	**1**	**3/10**	**-1/10**	**9/5**	
1	x_1	**1**	**0**	**-1/4**	**1/4**	**7/2**	
	Z_j	**1**	**1**	**1/20**	**3/20**		
	$\overline{cj} = C_j - Z_j$	**0**	**0**	**-1/20**	**-3/20**		

$x_2 = 9/5$; $x_1 = 7/2$; $Z_{max} = 5.3$

Since only x_1 is integer constrained, the problem is branched into two sub problems, each using one of the following additional constraints:

$x_1 \leq [\frac{7}{2}]$ or $x_1 \leq 3$, and

$x_1 \geq [\frac{7}{2}] + 1$ or $x_1 \geq 4$.

Optimal solution to the problem

Maximise $z = X_1 + X_2$

Subject to $2x_1 + 5x_2 \leq 16$,

$6x_1 + 5x_2 \leq 30$,

$x_2 \geq 0$,

$x_1 \geq 3$

is obtained in table 2 as $x_1 = 3$ and $x_2 = 2$, giving $Z_{max} = 5$. Since it satisfies the condition of x_1 being integer, it is the best solution available so far, and lower bound $Z_L = 5$.

	C_j	1	1	0	0	0		
C_B	**Basis**	X_1	X_2	S_1	S_2	S_3	**b_i**	**0**
0	S_1	2	5	1	0	0	16	8
0	S_2	6	5	0	1	0	30	5
0	S_3	(1)	0	0	0	1	3	3 ←
	Z_j	0	0	0	0	0		
	$\overline{c_j} = C_j - Z_j$	1	1	0	0	0		
		↑						
0	S_1	0	(5)	1	0	-2	10	2 ←
0	S_2	0	5	0	1	-6	12	12/5
1	x_1	1	0	0	0	1	3	3
	Z_j	1	0	0	0	1		
	$\overline{c_j} = C_j - Z_j$	0	1 ↑	0	0	-1		
1	x_2		0	1	1/5	0	- 2/5	2
0	S_2		0	0	-1	1	-4	2
1	x_1		1	0	0	0	1	3
	Z_j		1	1	1/5	0	3/5	
	$\overline{c_j} = C_j - Z_j$		0	0	-1/5	0	- 3/5	

$x_1 = 3$, $x_2 = 2$ and $Z_{max} = 5$.

Optimal solution to the problem

Maximise $z = X_1 + X_2$

Subject to $2x_1 + 5x_2 \leq 16$,

$6x_1 + 5x_2 \leq 30$,

$x_1 \geq 4$,

$x_2 \geq 0$

is obtained in table 2 as $x_1 = 4$ and $x_2 = 6/5$, which gives $Z_{max} = 26/5 = 5.2$. This solution also satisfies the condition of x_1 being non-negative integer and value of $Z = 5.2$ is better than the lower bound. Therefore this branch is also fathomed. Therefore, the optimal solution to the given problem is $x_1 = 4$ and $x_2 = 6/5$ and $Z_{max} = 5.2$.

Table 3

	C_j	1	1	0	0	0	**-M**		
C_B	**Basis**	X_1	X_2	S_1	S_2	S_3	**A**	b_i	**0**
0	S_1	2	5	1	0	0	0	16	8
0	S_2	6	5	0	1	0	0	30	5
-M	A	(1)	0	0	0	-1	1	4	4 ←
	Z_j	-M	0	0	0	M	-M		
	$\overline{c_j} = C_j - Z_j$	1+M	1	0	0	-M	0		
		↑							
0	S_1	0	5	1	0	2		8	8/5
0	S_2	0	(5)	0	1	6		6	6/5 ←
1	X_1	1	0	0	0	-1		4	-
	Z_j	1	0	0	0	-1			
	$\overline{c_j} = C_j - Z_j$	0	1	0	0	1			
			↑						
0	X_1	0	0	1	-1	-4		2	
1	X_2	0	1	0	1/5	6/5		6/5	
1	X_1	1	0	0	0	-1		4	
	Z_j	1	1	0	1/5	1/5			
	$\overline{c_j} = C_j - Z_j$	0	0	0	-1/5	-1/5			

$X_1 = 4$; $X_2 = 6/5$; $Z_{max} = 26/5 = 5.2$.

2.7 EXERCISE

Q1. Write short note on integer programming model.
Q2. Define and briefly explain I.P.P and mixed I.P.P.
Q3. What is the concept involved in Gomory's cutting plane method?
Q4. Describe a method of solving mixed I.P.P.
Q5. Explain the branch and bound method in integer programming.
Q6. Explain some of the practical applications of integer programming.

SOLVE THE FOLLOWING

1. Consider the problem of assigning three jobs to three men. Each man is capable of doing all the jobs; however, the time taken by the different men on each job is different and can be assumed to be known. The assignment has to be done so that each job is assigned only once, each man gets only one job and the total time taken by all the job is minimised. Formulate it as an I.P. problem with decision variables defined as
x_{ij} = [1, if the ith man is assigned to job j
0, otherwise.

2. A sales representative of a pharmaceutical company has been assigned a region and he must visit n cities in this region once in a quarter, starting from and returning to the regional headquarters. Formulate this as an I.P. problem to minimise the distance travelled.

3. A refrigeration and air-conditioning company has been awarded a contract for the air-conditioning of a new computer installation.

The company has to make a choice between two alternatives:
a. hire one or more refrigeration technicians for six hours a day or
b. hire one or more part-time refrigeration apprentice technicians for four hours a day.

The rate of wages of a refrigeration technician is Rs. 20 per hour, while the corresponding rate of apprentice technician is Rs. 8 per hour. The company wants to engage the technicians on work for not more than 25 man hours per day and also limit the charges to technicians to Rs. 440. The company estimates, that the productivity of a refrigeration technician is eight units and that of a part-time apprentice technician is three units. Formulate the integer programming problem to enable the company to select the optimum number of technicians and apprentices.

4. Solve the problem by Gomory's algorithm:

Maximise $Z = 3x_1 + 4x_2,$

subject to $x_1 + x_2 \leq 4,$

$3/5\, x_1 + x_2 \leq 3,$

$x_1, x_2 \geq 0$ and integer.

5. Solve by cutting plane algorithm:

Maximise $Z = 7x_1 + 10x_2,$

subject to $-x_1 + 3x_2 \leq 6,$

$7\, x_1 + x_2 \leq 35,$

$x_1, x_2 \geq 0$ and integer.

❖❖❖❖

3

GOAL PROGRAMMING

Unit Structure

3.1 INTRODUCTION:

As observed earlier, traditional mathematical programming models, including linear programming and integer programming, are based on the assumption that the decision making has a single, quantifiable, objective such as maximisation of profit or minimisation of inefficiency or cost. However, often there are situations, where instead of posing a single objective, managers use multiple criteria in decision-making. Thus, instead of setting only one objective, multiple goals may be set. Specifications of multiple goals creates difficulties in the solution to a given problem because the objectives are usually conflicting and incommensurate. For example, increasing a portfolio's annual return calls for making more risky investments. Thus, a portfolio manager's objective to maximise returns would conflict with minimising risk. Further, objectives may be incommensurate if they involve different units. Thus, maximisation of profits, expressed in monetary terms is an economic objective; maintaining a certain workforce level is an employment objective which is input in terms of the number of workers employed, while an environmental objective such as pollution may be measured in terms of damage to surroundings.

Goal Programming allows handling such multi-objective situations. It uses the concept of penalties in the format of linear programming. To apply goal programming, first a target value, a goal, is set for each of the objectives/goals. Since all the goals are unlikely to be satisfied simultaneously, a penalty is assigned for each unit of deviation from the target value in each direction. Thus, a revised linear programming problem, using 'deviational' variables is formulated whose optimal solution comes as close as possible to achieving the stated goals in the sense that it minimises the sum of penalties incurred for under-achieving or over-achieving the various goals. Thus, the central concept of goal programming is to determine the individual preferences of the decision makers in terms of the goals stated and to establish a single (an overall) function, which is to be minimised or a series of goal functions which are to be minimised in order of their relative importance.

3.2 GOAL PROGRAMMING WITH A SINGLE GOAL

Although goal programming is meant to deal with problems with multiple goals, we may begin with a problem with a single goal to develop some basic ideas.

EXAMPLE 1

A firm is producing two products which yield unit profits of Rs. 40 and Rs. 35 respectively. The two products are known to need 2 kg and 4 kg of raw material, respectively, per unit and 3 hours of labour each. With an availability of 60 kg of raw material and 96 labour hours, the problem is restated as below:

Maximise $Z = 40x_1 + 35x_2$

Subject to $2x_1 + 3x_2 \leq 60$

$4x_1 + 3x_2 \leq 96$

$x_1, x_2 \geq 0$

Now, suppose the manager has set the goal to achieve a profit of Rs. 1400. If we return to the graphic solution to this problem, we observe that the maximum profit obtainable in this case is Rs. 1000, and the profit represented by the Rs. 1400 iso-profit line cannot be realised. Thus, the problem is not feasible in terms of this requirement. However, goal programming, like managers, recognises that any goal stated may be under-achieved, met exactly, or over-achieved. For example, the goal of a profit of Rs. 1400 in the above situation would be under-achieved, while if the goal stated in the question is to attain a profit of Rs. 1000, it would be met exactly. On the other hand, if the manager sets to achieve a profit of Rs. 800, it would be over-achieved. Continuing with the manager's goal of achieving a profit of Rs. 1400, the requirement can be expressed using the mathematical abbreviation $40x_1 + 35x_2 = 1400$ as a goal. The implication is that the goal is to achieve the stated equality. (incidently, if the manager's goal was instead to have a profit of at least Rs. 1400, the goal would be expressed as $40x_1 + 35x_2 \geq 1400$). However, what is to be recognised is that such equalities or inequalities, which involve goals are not like inequalities related to the resources or other limitations. The inequalities (or equalities) related to resource capacity (like raw material or labour) or non-goal oriented limitations (like market size of a product), known as technological or structural constraints, are rigid while inequalities representing goal constraints are not so and are instead flexible.

We may now put the goal constraint into standard mathematical notation. In this particular case, we are aware that the goal constraint will be under-achieved. If we let d be the deviational variable of under-achievement, we can write $40x_1 + 35x_2 + d^- = 1400$. However, in a given problem, we would not know whether a goal will be under-achieved or be over-achieved or exactly met, we introduce two deviational variables - one of under-achievement and other of over-achievement. Thus, if we designate the deviational variable of over-achievement by d^+, we can express the above goal constraint as

$$40x_1 + 35x_2 + d^- - d^+ = 1400$$

where deviational variable d^- and d^+ are (i) either positive or zero and (ii) either d^- and d^+ is zero or (iii) both d^- and d^+ are zero. This relationship can always be met. For example, $x_1 = 0$, $x_2 = 0$, $d^- = 1400$ and $d^+ = 0$ provides a solution.

It may be noted here that when a goal is met exactly, both the deviational variables d^- and d^+ would be equal to zero. Further, we are assured that we never get a solution in which d^- and d^+ are both non-zero, since the objective is to minimise their sum-the combination that minimises will always have one of them at a zero level.

Thus, flexible goal constraints are always expressed in the way the given constraint has been shown above, by introducing deviational variables.

Now, reconsidering the example, we observe that we have two resource constraints and one converted goal constraint. To find the optimal solution, we first convert this problem to a standard minimisation or maximisation problem. This is effected by the concept of goal function. It is stipulated that we want to be as close to the profit of Rs. 1400 as possible and, therefore, minimise d^-, the under-achievement. Accordingly, the problem may be set as follows:

Minimise $Z = d$

Subject to $2x_1 + 3x_2 \leq 60$

$4x_1 + 3x_2 \leq 96$

$40x_1 + 35x_2 + d^- - d^+ = 1400$

$x_1, x_2, d^-, d^+ \geq 0$

We may solve this LPP to get $x_1 = 18$, $x_2 = 8$, $d^- = 400$ and $d^+ = 0$ and profit = Rs. 1000, implying thereby that the profit goal is under-achieved by Rs. 400 and that the maximum profit is Rs. 1000. The solution is the same as that of the original problem. The solution to the problem is contained in Tables 1, 2 and 3.

Table 1 Simplex Tableau 1: Non-optimal Solution

Basis		**X_1**	**X_2**	**S_1**	**S_2**	**D^-**	**D^+**	**B_i**	**B_i/a_{ij}**
S_1	**0**	2	3	1	0	0	0	60	30
S_2	**0**	4*	3	0	1	0	0	96	24
D^-	**1**	40	35	0	0	1	-1	1400	35 ←
C_j		0	0	0	0	0	0		
Solution		0	0	60	96	1400	0		
Δj		-40	-35	0	0	0	1		
		↑							

Table 2 Simplex Tableau 2: Non-optimal Solution

Basis		**X_1**	**X_2**	**S_1**	**S_2**	**D^-**	**D^+**	**B_i**	**B_i/a_{ij}**
S_1	**0**	0	3/2*	1	-1/2	0	0	12	8 ←
X_1	**0**	1	3/4	0	1/4	0	0	24	32
D^-	**1**	0	5	0	-10	1	-1	440	88
C_j		0	0	0	0	1	0		
Solution		24	0	12	0	440	0		
Δj		0	-5	0	10	0	1		
			↑						

Table 3 Simplex Tableau 3: Non-optimal Solution

Basis		**X_1**	**X_2**	**S_1**	**S_2**	**D^-**	**D^+**	**B_i**
X_2	**0**	0	1	2/3	-1/3	0	0	8
X_1	**0**	1	0	-1/2	1/2	0	0	18
D^-	**1**	0	0	-10/3	-25/3	1	-1	400
C_j		0	0	0	0	1	0	
Solution			8	0	0	400	0	
Δj		18	0			0	1	
		0	$\uparrow$	10/3	25/3			

3.3 GOAL PROGRAMMING WITH MULTIPLE GOALS

We now consider the usual goal programming situations which involve multiple goals. As indicated earlier, the basic approach of goal programming is to establish goal or target (in quantitative terms) for each objective, formulate an objective function for each objective and then look for a solution that minimises the deviations of these objective functions from their respective goals.

The goals that one may set may be either one-sided or two-sided. Further, one-sided goal can be a lower, which sets a lower limit that we do not want to fall under (but exceeding the limit is all right) or an upper, which sets an upper limit that we do not want to exceed (and falling short of which is just fine). Similarly, a two-sided goal sets a specific target, missing which from either side is not desired.

The deviational variables for any goal constraint measures the amount of discrepancy between the value of the objective and the goal specified in respect thereof.

Depending upon how the goals compare, goal programming problems may be categorised as being non-preemptive and preemptive. Non-preemptive goal programming involves situations where all the goals are of roughly comparable importance, whereas preemptive goal programming deals with cases where there is a hierarchy of priority levels for various goals so that there are goals of primary importance which receive attention before others which are of secondary importance. Thus, different goals are considered one by one according to their relative importance. We shall discuss the two types in turn.

3.4 NON-PREEMPTIVE GOAL PROGRAMMING

In non-preemptive goal programming, we first establish a goal for each objective and then seek a solution that minimises the sum of the deviation of these objectives from their respective goals. The goal programming assumes that the decision-maker has a linear utility function with respect to the objectives, that is to say, the marginal rate of substitution between the objectives is linear, regardless of the extent of deviation from the goal. Further, the deviations may be given different weights, called penalty weights, in accordance with the relative significance of the objectives, and the solution sought may be the one which minimises the weighted sum of the deviations. The weights used in a goal programming model are indicative of the decision-maker's utility for the various objectvies. Specifically, they measure the marginal rate of substitution between objectives and the degree of importance in attaining the goal of each objective relative to the others.

We shall illustrate the non-preemptive goal programming with the help of some examples. Consider the following:

EXAMPLE 2

The production manager of a company wants to schedule a week's production run for two products P_1 and P_2, each of which requires the labour and materials as shown below:

	Product	
	P_1	P_2
Labour Hours	2	4
Material M_1 (kg)	4	5
Material M_2 (kg)	5	4

The weekly availability of resources is limited to 600 labour hours, 1000 kg of material M_1 and 1200 kg of M_2. The unit profit for P_1 and P_2 is Rs. 20 and Rs. 32 respectively.

Products P_1 and P_2 are in fact, new models and are replacements of the older ones which have been discontinued very recently. The manager would like to maximise profit but he is equally concerned with maintaining workforce of the division at nearly constant level in the interest of employee morale. The workforce, which consists of people engaged in production, sales, distribution, peons and other general staff, consisted of 108 person in all. From a detailed study, it is known that production of one unit of P_1 would maintain 0.3 person in the workforce, while one unit of P_2 would maintain 0.75 person.

Had the production manager been considering only maximising profit, without regard to maintaining the workforce, he would do so by producing 167.67 units of P_1 and 66.67 units of P_2 (this can be checked by solving the problem as an LPP). On the basis of the available capacity, this would yield a profit equal to 167.67 x 20 + 66.67 x 32 or Rs. 5, 486.67. However, this would maintain 100.3 people in the workforce. The manager feels that probably he could increase the workforce requirement to desired level by accepting somewhat lower profit. In keeping with this, the following two goals are established: a profit of Rs. 5,400 per work and a workforce of 108 persons. Formulate and solve this as a goal programming problem.

If we let x_1 and x_2 represent the number of units of P_1 and P_2 respectively, to be produced every week, the goals and constraints of the problem can be stated as follows:

$$\begin{array}{ll}
20\,x_1 + 32\,x_2 = 5400 & \text{Goal 1} \\
0.3\,x_1 + 0.75\,x_2 = 108 & \text{Goal 2} \\
2x_1 + 4\,x_2 \le 600 & \text{Labour} \\
4x_1 + 5x_2 \le 1000 & \text{Material } M_1 \\
5x_1 + 4\,x_2 \le 1200 & \text{Material } M_2
\end{array}$$

This problem may not have a feasible solution which satisfies both the goals. Further, to be able to solve this problem using simplex method, we need to introduce deviational variables in each of the constraints involving goals. They may be defined as follows:

let d_1^- = number of rupees below the profit goal of Rs. 5400
let d_1^+ = number of rupees above the profit goal of Rs. 5400
let d_2^- = number of people below the workforce goal of Rs. 108
let d_2^+ = number of people above the workforce goal of Rs. 108

Now, considering the nature of goals, it may be presumed that over achievement of either of them would attract no penalty, and we may seek a solution which would minimise the under achievement of both of them. Accordingly, the problem may be stated as follows:

Minimise $Z = d_1^- + d_2^-$

Subject to

$$20\,x_1 + 32\,x_2 + d_1^- - d_1^+ = 5400$$
$$0.3\,x_1 + 0.75\,x_2 + d_2^- - d_2^+ = 108$$
$$2x_1 + 4\,x_2 + S_1 = 600$$
$$4x_1 + 5x_2 + S_2 = 1000$$
$$5x_1 + 4\,x_2 + S_3 = 1200$$
$$x_1, x_2, S_1, S_2, S_3, d_1^-, d_1^+, d_2^- \text{ and } d_2^+ \geq 0$$

It may be observed here that, as discussed earlier, the goals are written as constraints. Further, in solving a problem by simplex method, while artificial variables are ordinarily introduced when equations are involved, it is not so in the context of the constraints involving goals. Here, expressing the deviation from goal as the difference between two non-negative variables ($d_1^- - d_2^+$, for example) provides the model sufficient flexibility to permit the under achievement or over achievement of the goal. To solve the problem using simplex method, the initial solution will be provided by the two deviational variables d_1^- and d_2^- and the three slack variables S_1, S_2 and S_3 introduced in the three constraints involving resources: labour and materials. The detailed solution to the problem is contained in Tables 1 and 2.

The optimal solution to the problem as obtained from Table 1 is: $x_1 = 150$, $x_2 = 75$, $d_2^- = 27/4 = 6.75$, $S_2 = 25$ and $S_3 = 150$, with the objective function value equal to 6.75. From this, it is evident that workforce shall be 101.25, with the employment goal being under achieved to the extent of 6.75 people, while 25 kg of material M_1 and 150 kg of material M_2 would remain unutilised. The other variables are non-basic so that all the available labour hours shall be used and the profit goal be met exactly.

TABLE 1 Simplex Tableau 1

Basis	**X_1**	**X_2**	**d_1^-**	**d_1^+**	**d_2^-**	**d_2^+**	**S_1**	**S_2**	**S_3**	**B_i**	**B_i/a_{ij}**
d_1^- 1	20	32	1	-1	0	0	0	0	0	5400	168.75
d_2^- 1	3/10	3/4*	0	0	1	-1	0	0	0	108	144 ←
S_1 0	2	4	0	0	0	0	1	0	0	600	150
S_2 0	4	5	0	0	0	0	0	1	0	1000	200
S_3 0	5	4	0	0	0	0	0	0	1	1200	300
C_j	0	0	1	0	1	0	0	0	0		
Solution	0	0	5400	0	108	0	600	1000	1200		
Δj	-213/10	-131/4	0	0	1	0	0	0	0		

TABLE 2 Simplex Tableau 2

Basis	X_1	X_2	d_1^-	d_1^+	d_2^-	d_2^+	S_1	S_2	S_3	B_i	B_i/a_{ij}	
d_1^- **1**	36/5	0	1	-1	-128/3	128/3	0	0	0	792	299/16	
X_2 **0**	2/5	1	0	0	4/3	4/3	0	0	0	144	-	←
S_1 **0**	2/5	0	0	0	-16/3	16/3*	1	0	0	24	9/2	←
S_2 **0**	2	0	0	0	-20/3	20/3	0	1	0	280	42	
S_3 **0**	17/5	0	0	0	-16/3	16/3	0	0	1	624	117	
C_j	0	0	1	0	1	0	0	0	0			
Solution	0	144	792	0	0	0	24	280	624			
Δj	-36/5	0	0	1	131/3	-128/3	0	0	0			
						↑						

TABLE 3 Simplex Tableau 3

Basis	X_1	X_2	d_1^-	d_1^+	d_2^-	d_2^+	S_1	S_2	S_3	B_i	B_i/a_{ij}	
d_1^- **1**	4	0	1	-1	0	0	-8	0	0	600	150	
X_2 **0**	1/2	1	0	0	0	0	1/4	0	0	150	300	
d_2^+ **0**	3/40*	0	0	0	-1	1	3/16	0	0	9/2	60	←
S_2 **0**	3/2	0	0	0	0	0	-5/4	1	0	250	500/3	
S_3 **0**	3	0	0	0	0	0	-1	0	1	600	200	
C_j	0	0	1	0	1	0	0	0	0			
Solution	0	150	600	0	0	9/2	0	250	600			
Δj	-4	0	0	1	0	0	8	0	0			
	↑											

TABLE 4 Simplex Tableau 4

Basis	X_1	X_2	d_1^-	d_1^+	d_2^-	d_2^+	S_1	S_2	S_3	B_i	B_i/a_{ij}	
d_1^- **1**	0	0	1	-1	160/3*	-160/3	-18	0	0	360	27/4	←
X_2 **0**	0	1	0	0	20/3	-20/3	-1	0	0	120	18	
X_1 **0**	1	0	0	0	-40/3	40/3	5/2	0	0	60	-	
S_2 **0**	0	0	0	0	20	20	-5	1	0	160	8	
S_3 **0**	0	0	0	0	40	40	-17/2	0	1	420	21/2	
C_j	0	0	1	0	1	1	0	0	0			
Solution	60	120	360	0	0	0	0	160	420			
Δj	0	0	0	1	-157/3	-157/3	18	0	0			
						↑						

TABLE 5 Simplex Tableau 5

Basis		X_1	X_2	d_1^-	d_1^+	d_2^-	d_2^+	S_1	S_2	S_3	B_i
D_2^-	**1**	0	0	3/160	-3/160	1	-1	-27/80	0	0	27/4
X_2	**0**	0	1	-1/8	1/8	0	0	5/4	0	0	75
X_1	**0**	1	0	1/4	-1/4	0	0	-2	0	0	150
S_2	**0**	0	0	-3/8	3/8	0	0	7/4	1	0	25
S_3	**0**	0	0	-3/4	3/4	0	0	5	0	1	150
C_j		0	0	1	0	1	0	0	0	0	
Solution		150	75	0	0	27/4	0	0	25	150	
Δj		0	0	157/160	3/160	0	1	27/80	0	0	

3.5 MODIFIED SIMPLEX METHOD FOR GOAL PROGRAMMING

Pre-emptive goal programming problems can be solved using the simplex method in a modified form. Before illustrating the application of the method for solving such problems, let us recall that in such problems, the objective is to minimise the unattained portion of the goals, which is sought to be achieved by assigning pre-emptive priority factors P_1, P_2, ...P_n and so on (and differential weighting in case of equal priority goals) to the deviational variables introduced in respect of various goals. These serve to act as the c_j values. However, it is significant to note that the priority factors are only ordinal weights (that is to say, they can be ranked in order of their importance). They are not commensurable and, as such, no values can be assigned to them. The Δj values here are not expressed (like in case of linear programming) in a single row and are written instead, in the form of a matrix, with the number of rows equal to pre-emptive priority factors and the number of columns being equal to the number of variables involved. The expression of values as a matrix calls for a modified way of selecting the key column to identify the incoming variable. Since $P_j >>> P_{j+1}$, the selection procedure involves initiation from P_j and moving to lower priority goals.

Let us consider the following example:

EXAMPLE 1:

Solve the goal programming problem:

Minimise $Z = P_1d_1^- + P_2(4d_2^- + 3d_3^-) + P_3d_1^+$

Subject to $X_1 + X_2 \leq 300$

$2X_1 + X_2 + d_1^- - d_1^+ = 400$

$X_1 + d_2^- = 150$

$X_2 + d_3^- = 350$

X_1, X_2, d_1^-, d_2^- and $d_3^- \geq 0$

To solve this problem, we first express the given information as shown in Table 1. Observe that the basic variables in Simplex Tableau are S_1, d_1^-, d_2^- and d_3^- with solution values equal to 300, 400, 150 and 350 respectively. Evidently, since both X_1 and X_2 are equal to zero, the zero output implies various negative deviational variables d_1^-, d_2^- and d_3^- are assuming positive values. The rows under the c_j row are listed according to the priority levels of the goals, with the highest priority goal P_1 at the bottom and the lower ranked goals P_2 and P_3 placed above it in the ascending order. The values in these indicate the Δj values. Let us understand the calculation of Δj = (c_j - z_j) values in the table. For calculating z_j values, consider the weights (coefficients) mentioned with P_1, P_2 and P_3 etc. First, look to the column beaded x_1. Since the value in this column corresponding to the row involving P_1 (basic variable d_1^+) is 2, we get $z_j = 2 \times 1 = 2$ in respect of P_1. With $c_j = 0$ for x_1, the $\Delta j = 0 - 2 = -2$ is written in the bottom row marked P_1. Now, for P_2, the values in the column are 1 and 0, with the corresponding basic values being 4 and 3, we get $z_j = 1 \times 4 + 0 \times 3 = 4$. With $c_j = 0$ for x_1, the $\Delta j = 0 - 4 = -4$ is shown in the row marked P_2. There being on P_3 in the basis the $\Delta j = 0$ in respect of the row marked P_3 under x_1. As another example, consider the column headed d_1^+. Since the value in this column opposite the basis variable d_1^- is -1, we get $z_j = 1 \times -1 = -1$. Since both the values in the column headed d_1^+ are zero, $z_j = 4 \times 0 + 0 \times 3 = 0$ and $\Delta j = 0 - 0 = 0$ against the row marked P_2. Finally, since no variable in the basis has the priority level P_3, we have $z_j = 0$ and $\Delta j = 1 - 0 = 1$.

The values against rows marked P_1, P_2 and P_3 under the column headed b_i and z_j values and they are derived as follows. The solution value of d_1^- (with priority level 1) is 400. With the weightage of P_1 being 1, the z-value is $400 \times 1 = 400$. Similarly, with $d_2^- = 150$ and $d_3^- = 350$ in the table and the weightage being 4 and 3 respectively, the z_j value for this priority goal works out to be 150 x 4 + 350 x 3 = 1650. Finally, there being no basic variable priority P_3, we have $d_1^+ = 0$ and z-value for this priority equal to 1 x 0 = 0.

3.6 TEST OF OPTIMALITY AND DERIVATION OF REVISED TABLEAU

To test the optimality, we first look to the Δj values in the bottom row, headed P_1. The objective function being of minimisation type, the solution is not optimal since all Δj 's are not greater than, or equal to zero. To obtain a revised simplex tableau, we consider the most negative of the Δj values in the P_1 row in the Simplex Tableau 1. It being equal to -2 for the common headed $x_{1,}$ the variable x_1 is the incoming variable. Using a_{ij} values in this column, we calculate replacement ratios b_i/a_{ij} and select the least non-negative one, which in this case is 150, corresponding to the variable d_2^-. Thus, d_2^- is taken to be the outgoing variable.

TABLE 1 Simplex Tableau 1: Non-optimal Solution

Basis		**X_1**	**X_2**	**S_1**	**d_1^-**	**D_1^+**	**d_2^-**	**D_3^-**	**B_i**	**B_i/a_{ij}**
S_1	**0**	1	1	1	0	-1	0	0	300	300
d_1^-	**P_1**	2	1	0	1	0	0	0	400	200
d_2^-	**$4P_2$**	1*	0	0	0	0	1	0	150	150 $\leftarrow$
D_3^-	**$3P_2$**	0	1	0	0	0	0	1	350	-
C_j		0	0	0	P_1	P_3	$4P_2$	$3P_2$		
Δj	$P3$	0	0	0	0	1	0	0	0	
	$P2$	-4	-3	0	0	0	0	0	1650	
	$P1$	-2	-1	0	0	1	0	0	400	
		$\uparrow$								

TABLE 2 Simplex Tableau 2: Non-optimal Solution

Basis		X_1	X_2	S_1	d_1^-	D_1^+	d_2^-	D_3^-	B_i	B_i/a_{ij}
S_1	**0**	0	1	1	0	0	-1	0	150	150
d_1^-	P_1	0	1*	0	1	-1	-2	0	100	100 ←
X_1	**0**	1	0	0	0	0	1	0	150	-
D_3^-	$3P_2$	0	1	0	0	0	0	1	350	350
C_j		0	0	0	P_1	P_3	$4P_2$	$3P_2$		
Δj	P3	0	0	0	0	1	0	0	0	
	P2	0	-3	0	0	0	4	0	1050	
	P1	0	-1	0	0	1	2	0	100	
			↑							

TABLE 3 Simplex Tableau 3: Non-optimal Solution

Basis		X_1	X_2	S_1	d_1^-	D_1^+	d_2^-	D_3^-	B_i	B_i/a_{ij}
S_1	**0**	0	0	1	1	1*	-1	0	50	50 ←
X_2	**0**	0	1	0	1	-1	-2	0	100	-
X_1	**0**	1	0	0	0	0	1	0	150	-
D_3^-	$3P_2$	0	0	0	-1	1	2	1	250	250
C_j		0	0	0	P_1	P_3	$4P_2$	$3P_2$		
Δj	P3	0	0	0	0	1	0	0	0	
	P2	0	0	0	3	-3	-2	0	750	
	P1	0	0	0	1	0	0	0	0	
						↑				

TABLE 3 Simplex Tableau 3: Non-optimal Solution

Basis		X_1	X_2	S_1	d_1^-	D_1^+	d_2^-	D_3^-	B_i
D_1^+	P_3	0	0	1	-1	1*	1	0	50
X_2	**0**	0	1	1	0	-1	-1	0	150
X_1	**0**	1	0	0	0	0	1	0	150
D_3^-	$3P_2$	0	0	-1	0	1	1	1	200
C_j		0	0	0	P_1	P_3	$4P_2$	$3P_2$	
Δj	P3	0	0	-1	0	0	0	0	50
	P2	0	0	3	3	0	1	0	600
	P1	0	0	0	1	1	0	0	0

It may be mentioned here that if there is a tie in the most negative Δj values of a given priority level, we consider the corresponding Δj values in the next (lower) priority level and select the variable which has a greater numerical value (ignoring sign) of Δj.

Once the incoming and outgoing variables (key column and key row) are determined, the revised simplex tableau can be obtained in the same way as in case of linear programming problems. The resulting tableau is contained in Table 2. We have, output of prouct A, x_1 = 150; unused materials S_1 = 150 kg; under achievement of goal 2, d_3^- = 350, being the output of product B short by as many units of the goal. For this solution as well, the Δj values are calculated in the same way as discussed int he context of solution given in table 1. As there is a negative Δj value (= -1) for the variable x_2, it is taken to be the incoming variable. With replacement ratios calculated with the help of a_{ij} values of the key column, it is evident that d_1^- is the outgoing variable. The next solution is contained in table 3.

For the solution given in Table 3, it is evident that there are no negative Δj values in the row marked P_1, implying that optimality has been achieved in relation to the top priority goal. Hence, we consider the Δj values in the row marked P_2. Here, two negative values seen are -3 and -2 under d_1^+ and d_2^-, respectively and we select the more negative value of the two, that is, -3. Thus, the variable d_1^+ enters and S_1 leaves as the least non-negative replacement ratio corresponds to this variable.

Finally, the revised Simplex Tableau is given in Table 4 which gives the optimal solution. It is optimal in the sense that it enables the manager to attain the set goals as closely as possible within the given decision scenario and the priority structure. It may be noted that Δj's greater than or equal-to-zero in the rows marked P_1 and P_2. However, there is a negative Δj (= -1) under the variable S_3 in the P_3 row. Apparently, we can improve the solution by considering this negative Δj value by introducing S_1 in the solution. But, it may be observed that this can be done only at the expense of achieving the higher priority goal, as a positive Δj value, equal to 3, for this variable is present in the row P_2. The rule, then, is that if there is negative Δj value at the lower priority level, then the variable in that column cannot be selected as the incoming variable if there is a positive Δj value at a higher priority level.

From the simplex tableau 4, the optimal solution may be stated as follows: x_1 = 150; x_2 = 150; d_1^+ = 50 and d_3^- = 200, which implies that 150 units of each of the products A and B be produced so that the target of selling 350 units of product B shall be under achieved by 200 units and an over-time work of 50 hours shall be involved.

3.7 EXERCISES:

1. Differentiate between non-preemptive and preemptive goal programming problems. How is the solution method different in the two cases?

2. "In goal programming we attempt to 'satisfy' or come as close as possible to satisfying, the various goals". Discuss.

3. "Goal Programming appears to be the most appropriate, flexible and powerful quantitative technique for complex decision problems involving conflicting objectives." Discuss this statement.

SOLVE THE FOLLOWING

1. Write constraints to satisfy each of the following conditions in a project selection model. The projects are numbered 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

i. Exactly one project from the set (1,2,3) must be selected.

ii. Project 2 can be selected only if number 10 is selected. However, 10 can be selected without 2 being selected.

iii. No more than one project from the set (1,3,5,7,9) can be selected.

iv. If number 4 is selected, then number 8 cannot be selected.

v. Projects 4 and 10 must both be selected or both be rejected.

2. A manufacturer of toys makes two types of toys, A and B, processing of these two toys is done on two machines X and Y. Toy A requires 4 hours on machine X and six hours on machine Y, while Toy B requires eight hours on machine X and five hours on machine Y. There are thirty two hours of time per day available on machine X and thirty hours on machine Y. The profit obtained on each of the toys is Rs. 30 per unit. What should be the daily

production of each of the toys for maximum profit? A non-integr solution to the problem is not acceptable.

3. A rural clinic hires its staff from nearby cities and towns on a part-time basis. The clinic attempts to have a general practitioner (GP), s nurse and an internist on duty during at least a portion of each week. The clinic has a weekly budget of Rs. 1, 200. A GP charges the clinic Rs. 40 per hour, a nurse charges Rs. 20 per hour, and an internist charges Rs. 150. The clinic has established the following goals in order of priority.

i. A nurse should be available for at least 30 hours per week.

ii. The weekly budget of Rs. 1, 200 should not be exceeded.

iii. A GP or internist should be available at least 20 hours per week.

iv. An internist should be available at least 6 hours per week.

Formulate a goal programming model for determining the number of hours to hire each staff member in order to satisfy the various goals.

4. The Super-Star Company produces three models of microcomputers - SS100, SS150 and SS250. Most of the components are imported from the Far East, and the company only assembles the microcomputers. The operation hours required to produce one unit of the microcomputers are 2 hours for SS100, 3 hours for SS150 and 4 hours for SS250. The normal capacity of the assembly line is 400 hours per month. The profits per unit are Rs. 10, 000 for SS100, Rs. 15, 000 for SS150 and Rs. 25, 000 for SS250. The President of the company has set the following goals, according to their order of importance.

i. Avoid the under utilisation of production capacity.

ii. Meet the outstanding orders - 30 units for SS100, 20 units for SS150 and 50 units for SS250.

iii. Avoid the over utilisation of the production capacity.

iv. Maximise total profit as much as possible.

Formulate a goal programming model for the problem and solve it (up to 3 iterations only).

❖❖❖❖

4

PARAMETRIC PROGRAMMING

Unit Structure

4.1 INTRODUCTION

Parametric Linear Programming investigates the effect of predetermined continuous variations of these coefficients on the optimal solution. It is simply an extension of sensitivity analysis and aims at finding the various basic solutions that become optimal, one after the other, as the coefficients of the problem change continuously. The coefficients change as a linear function of a single parameter, hence the name parametric linear programming for this computational technique. As in sensitivity analysis, the purpose of this technique is to reduce the additional computations required to obtain the changes in the optimal solution. The various types of parametric problems that one may come across are:

i. Parametric Cost Problem: In which the cost coefficients c_j vary linearly as a function of parameter λ.

ii. Parametric right-hand side problem: In which the requirement coefficients b_i vary linearly as a function f parameter λ.

iii. Parametric problem involving linear variations in the non-basic vector P_j of A.

iv. Parametric Problem involving simultaneous linear variations in c_j, b_i and P_j.

This text covers type 1 and type 2 parametric problems in details.

4.2 PARAMETRIC COST PROBLEM

Let the linear programming problem before parametrisation be
Minimise $Z = CX$,
subject to $AX = b$
$X \geq 0$,
where C is the given cost vector.

Let this cost vector change to $C + \lambda.C'$ so that the parametric cost problem becomes
minimise $Z = (C + \lambda C')X$,
subject to $AX = b$,
$X \geq 0$,
where is the given predetermined cost variation vector and λ is an unknown (positive or negative) parameter. As λ changes, the cost coefficients of all variables also change. We wish to determine the family of optimal solution as λ changes from $-\infty$ to $+\infty$.

This problem is solved by using the simplex method and sensitivity analysis. When $\lambda = 0$, the parametric cost problem reduces to the original L. P. Problem; simplex method is used to find its optimal solution. Let B and X_B represent the optimal basis matrix and the optimal basic feasible solution respectively for $\lambda = 0$. The net evaluation or relative cost coefficients are all non-negative (minimisation problem) and are given by

$$\overline{C_J} = c_j - Z_j = c_j - \Sigma\, c_B a_{ij} = c_j - c_B \overline{P_J},$$

where c_B is the cost vector of the basic variables and $\overline{P_J}$ is the jth column (corresponding to the variable x_j) in the optimal table.
As λ changes from zero to a positive or negative value, the feasible region and values of the basic variables X_B remain unaltered, but the relative cost coefficients change. For any variable x_j, the relative cost coefficient is given by

$$\overline{c_J}(\lambda) = (c_j + \lambda c_j') - (c_B + \lambda c'_B)\overline{P_J} = (c_j - c'_B\overline{P_J}) + \lambda(c_j' - c'_B\overline{P_J}) = \overline{C_J} + \lambda\overline{c'_J},$$

Since vectors $\dot{C}$ and C' are known, $\overline{c_J}$ and $\overline{c'_J}$ can be determined. For the current minimisation problem, $\overline{c_J}$ (λ) must be non-negative for the solution to be optimal $[\overline{c_J}(\lambda)$ must be non-positive for a maximisation problem]. Thus
$\overline{c_J}(\lambda) \geq 0$, $c_j + \lambda c_j' \geq 0$.

In other words, for a given solution we can determine the range λ within which the solution remains optimal.

EXAMPLE 1

Consider the linear programming problem

maximise $Z = 4x_1 + 6x_2 + 2x_3,$

subject to $x_1 + x_2 + x_3 \leq 3,$

$x_1 + 4x_2 + 7x_3 \leq 9$

$x_1, x_2, x_3 \geq 0.$

The optimal solution to this problem is given by the following table:

Table 1

	C_j	4	6	2	0	0	
C_B	**Basis**	X_1	X_2	X_3	X_4	X_5	**b**
4	X_1	**1**	**0**	**-1**	**4/3**	**-1/3**	**1**
6	X_2	**0**	**1**	**2**	**-1/3**	**1/3**	**2**
$Z_j = \Sigma\ c_B\ a_{ij}$		**4**	**6**	**8**	**10/3**	**2/3**	
$\overline{cj} = c_j - Z_j$		**0**	**0**	**-6**	**-10/3**	**-2/3**	

Solve this problem if the variation cost vector C' = (2, -2, 2, 0, 0). Identify all critical values of the parameter λ.

Solution

The given parametric cost problem is

maximise $Z = (4 + 2\lambda)\ x_1 + (6 - 2\lambda)x_2 + (2 + 2\lambda)x_3,$

subject to $x_1 + x_2 + x_3 + x_4 = 3,$

$x_1 + 4x_2 + 7x_3 + x_5 = 9$

$x_1, x_2, x_3, x_4, x_5 \geq 0.$

When λ = 0, the problem reduces to the L.P. problem, whose optimal solution is given by Table 1. The relative profit coeffiecients in this optimal table are all non-positive. For values of λ other than zero, the relative profit coefficients become linear functions of λ. To compute them, we first, add a new relative profit row called $\overline{c'j}$ row to table 1. This is shown in table 2.

TABLE 2

		C'_j	2	-2	2	0	0	
		$\mathbf{C_j}$	**4**	**6**	**2**	**0**	**0**	
C'_B	$\mathbf{C_B}$	**Basis**	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$	$\mathbf{X_4}$	$\mathbf{X_5}$	**b**
2	**4**	$\mathbf{X_1}$	**1**	**0**	**-1**	**4/3**	**-1/3**	**1**
-2	**6**	$\mathbf{X_2}$	**0**	**1**	**2**	**-1/3**	**1/3**	**2**
	$\overline{cj}$		**0**	**0**	**-6**	**-10/3**	**-2/3**	**Z = 16**
	$\overline{c'j}$		**0**	**0**	**8**	**-10/3**	**4/3**	**Z' = -2**

In table 2, $\overline{c'j}$ is calculated just as $\overline{cj}$ row except that vector C is replaced by C'. For example,

$\overline{c2}$ = $c_2 - z_2 = c_2\ \Sigma\ C_B\ a_{i2} = c_2 - C_B\ \overline{P2}$ = $6 - (4, 6)\begin{matrix}0\\1\end{matrix} = 6 - 6 = 0.$

$\overline{C'1} = c'_1 - C'BPT = 2 - (2, 2)\begin{matrix}1\\0\end{matrix} = 0.$

$\overline{c'2} = -2 - (2, -2)\begin{matrix}0\\1\end{matrix} = 0.$

$\overline{c'3} = 2 - (2, -2)\begin{matrix}-1\\2\end{matrix} = 2 - (-2 -4) = 8,$

$\overline{c'4} = 0 - (2, -2)\begin{matrix}\frac{4}{3}\\-\frac{1}{3}\end{matrix} = - [8/3 + 2/3) = -10/3$

$\overline{c'5} = 0 - (2, -2)\begin{matrix}-\frac{1}{3}\\\frac{1}{3}\end{matrix} = - [-2/3 - 2/3] = 4/3.$

Z' = 1 X 2 - 2 X 2 = -2.

Table 2 represents a basic feasible solution for the given parametric cost problem. It is given by

$x_1 = 1; x_2 = 2; x_3 = x_4 = x_5 = 0.$

Value of the objective function, $Z(\lambda) = Z + \lambda Z' = 16 - 2\lambda$

$\overline{cj}(\lambda) = \overline{cj} + \lambda\,\overline{c'j}$, j = 1, 2, 3, 4, 5.

Table 2 will be optimal if $\overline{cj}(\lambda) \leq 0$ for j = 3,4,5. Thus we can determine the range of λ for which table 2 remain optima as follows:

$\overline{c3}(\lambda) = \overline{c3} + \lambda\overline{c'3} = -6 + 8\lambda \leq 0$ or $\lambda \leq 3/4$,

$\overline{c4}(\lambda) = \overline{c4} + \lambda\overline{c'4} = -10/3 - 10/3\lambda \leq 0$ or $\lambda \leq -1$,

$\overline{c5}(\lambda) = \overline{c5} + \lambda\overline{c'5} = -2/3 + 4/3\lambda \leq 0$ or $\lambda \leq 1/2$.

Thus $x_1 = 1$, $x_2 = 2$; $x_3 = x_4 = x_5 = 0$ is an optimal solution for the given parametric problem for all values of λ between -1 and 1/2 and $Z_{max} = 16 - 2\lambda$.

For λ > 1/2, the relative profit coefficient of the non-basic variable x_5, namely $\overline{c5}(\lambda)$ becomes positive and table 2 no longer remains optimal. Regular simplex method is used to iterate towards optimality. X_5 is the entering variable and computation of 'θ' - column indicates X_2 to be the variable that leaves the basis matrix so that the key elements is 1/3. The key element is made unity and X_2 is replaced by X_5 in table 3.

TABLE 3

		C'_j	2	-2	2	0	0	
		$\mathbf{C_j}$	**4**	**6**	**2**	**0**	**0**	
C'_B	$\mathbf{C_B}$	**Basis**	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$	$\mathbf{X_4}$	$\mathbf{X_5}$	**b**
2	**4**	$\mathbf{X_1}$	**1**		**1**	**1**	**0**	**3**
0	**0**	$\mathbf{X_2}$	**0**	**3**	**6**	**-1**	**1**	**6**
		$\overline{c_j}$	**0**	**2**	**-2**	**-4**	**0**	**Z = 12**
		$\overline{c'_j}$	**0**	**-4**	**0**	**-2**	**0**	**Z' = -6**

Table 3 will be optimal if $\overline{c_j}$ (λ) ≤ 0, for j = 2, 3, 4.

Now, $\overline{c_2}$ (λ) = $\overline{c_2}$ + λ$\overline{c'_2}$ = 2 - 4 λ ≤ 0 or λ ≥ 1/2,

$\overline{c_3}$ (λ) = $\overline{c_3}$ + λ$\overline{c'_3}$ = -2 ≤ 0 which is true,

$\overline{c_4}$ (λ) = $\overline{c_4}$ + λ$\overline{c'_4}$ = -4 - 2λ ≤ 0 or λ ≥ -2.

□ For all λ ≥ 1/2, the optimal solution is given by x_1 = 3, x_2 = x_3 = x_4 0; x_5 = 6 and Z_{max} = 12 + 6λ.

For λ □ -1, the relative profit coefficient of the non-basic variable x_4 becomes the entering variable and x_1 the leaving variable. Key element is 4/3. This element is made unity and x_1 is replaced by x_4 in table 4.

TABLE 4

		C'_j	2	-2	2	0	0	
		$\mathbf{C_j}$	**4**	**6**	**2**	**0**	**0**	
C'_B	$\mathbf{C_B}$	**Basis**	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$	$\mathbf{X_4}$	$\mathbf{X_5}$	**b**
0	**0**	$\mathbf{X_4}$	**3/4**	**0**	**-3/4**	**1**	**-1/4**	**3/4**
-2	**6**	$\mathbf{X_2}$	**1/4**	**1**	**7/4**	**0**	**1/4**	**9/4**
		$\overline{c_j}$	**5/2**	**0**	**-17/2**	**0**	**-3/2**	**Z = 27/2**
		$\overline{c'_j}$	**5/2**	**0**	**11/2**	**0**	**1/2**	**Z' = -9/26**

Table 4 will be optimal if $\overline{c_j}$ (λ) ≤ 0, for j = 1, 3, 5.

Now $\overline{c_1}$ (λ) = **1** + λ$\overline{c'_1}$ = 5/2 + 5/2λ ≤ 0 or λ ≤ -1,

$\overline{c_3}$ (λ) = $\overline{c_3}$ + λ$\overline{c'_3}$ = -17/2 + 11/2 λ □ λ ≤ 17/11.

$\overline{c_5}$ (λ) = $\overline{c_5}$ + λ$\overline{c'_5}$ = -3/2 + 1/2 λ ≤ 0 or λ ≤ 3.

□ For all λ ≥ -1, the optimal solution is given by
x_1 = 0, x_2 = 9/4; x_3 = 0; x_4 = 3/4; x_5 = 0 and Z_{max} = 27/2 - 9/2λ.

Thus tables2,3 and 4 give families of optimal solutions for -1 ≤ λ ≤ 1/2, λ ≥ 1/2 and λ ≤ -1 respectively.

4.3 PARAMETRIC RIGHT-HAND SIDE PROBLEM

The right-hand side constants in a linear programming problem represent the limits in the resources and the outputs. In some practical problems all the resources are not independent of one another. A shortage of one resource may cause shortage of other resources at varying levels. Same is true for outputs also. For example, consider a firm manufacturing electrical appliances. A shortage in electric power will decrease the demand of all the electric items produced, in varying degrees depending upon the electric energy consumed by them. In all such problems, we are to consider simultaneous changes in the right-hand side constants, which are functions of one parameter and study how the optimal solution is affected by these changes.

Let the linear programming problem before parametrisation be
maximise $Z = cX$.
subject to $AX = b$,
$X = 0$,
where b is the known requirement (right-hand side) vector. Let this requirement vector b change to $b + \lambda b'$ so that parametric right-hand side problem becomes:
maximise $Z = cX$.
subject to $AX = b + \lambda b'$
$X \geq 0$,
where b' is the given and pre determined variation vector and λ is an unknown parameter. As λ changes, the right-hand constants also change. We wish to determine the family of optimal solution as λ changes from $-\infty$ to $+\infty$.

When $\lambda = 0$, the parametric problem reduces to the original L.P. problem; simplex method is used to find its optimal solution.

Let B and X_B represent the optimal basis matrix and the optimal basic feasible solution respectively for $\lambda = 0$. Then $X_B = B^{-1} b$. As λ changes from zero to a positive or negative value, the values of the basic variables change and the new values are given by

$$X_B = B^{-1} (b + \lambda b') = B^{-1} b + \lambda B^{-1} b' = \bar{b} + \lambda \bar{b'}.$$

A change in λ has no effect on the values of relative profit coefficients $\bar{c_j}$ i.e., $\bar{c_j}$ values remain non-positive (maximisation problem). For a given basis matrix B, values of $\bar{b}$ and $\bar{b'}$. can be calculated. The solution $X_B = \bar{b} + \lambda \bar{b}'$ is feasible and optimal as long as $\bar{b} + \lambda \bar{b}' \geq 0$. In other words, for a given solution we can determine the range for λ within which the solution remains optimal.

EXAMPLE 1

Consider the linear programming problem

maximise $Z = 4x_1 + 6x_2 + 2x_3,$

subject to $x_1 + x_2 + x_3 \leq 3,$

$x_1 + 4x_2 + 7x_3 \leq 9$

$x_1, x_2, x_3 \geq 0.$

The optimal solution to this problem is given by

Table 1

	C_j	4	6	2	0	0	
C_B	**Basis**	X_1	X_2	X_3	X_4	X_5	**b**
4	X_1	**1**	**0**	**-1**	**4/3**	**-1/3**	**1**
6	X_2	**0**	**1**	**2**	**-1/3**	**1/3**	**2**
$Z_j = \Sigma\, c_B a_{ij}$		**4**	**6**	**8**	**10/3**	**2/3**	
$\overline{C_j} = c_j - Z_j$		**0**	**0**	**-6**	**-10/3**	**-2/3**	

Solve the problem if the variation right-hand side vector $b' = \begin{matrix} 3 \\ -3 \end{matrix}$. Perform complete parametric analysis and identify all critical values of parameter λ.

Solution

The given parametric right-hand side problem is

maximise $Z = 4x_1 + 6x_2 + 2x_3 + 0\,x_4 + 0\,x_5,$

subject to $x_1 + x_2 + x_3 + x_4 = 3 + 3\lambda,$

$x_1 + 4x_2 + 7x_3 + x_5 = 9 - 3\lambda$

$x_1, x_2, x_3, x_4, x_5 \geq 0.$

When λ = 0, the problem reduces to the L. P. problem whose optimal solution is given by table 1. For values of λ other than 0, the values of right-hand constants change because of the variation vector b'. This is shown in the expanded table 2.

Table 2

	C_j	4	6	2	0	0		
C_B	**Basis**	X_1	X_2	X_3	X_4	X_5	$\bar{b}$	$\bar{b}'$
4	X_1	**1**	**0**	**-1**	**4/3**	**-1/3**	**1**	**5**
6	X_2	**0**	**1**	**2**	**-1/3**	**1/3**	**2**	**2** $\leftarrow$
	$\overline{c_j}$	**0**	**0**	**-6**	**10/3**	**-2/3**	**Z = 16**	**Z' = 8**
					$\uparrow$**K**			

The vector $\bar{b}$ and $\bar{b}$ are computed as follows:

$$\bar{b} = B^{-1} b = \begin{matrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} \end{matrix} \begin{matrix} 3 \\ 9 \end{matrix} = \begin{matrix} 1 \\ 2 \end{matrix}$$

$$\bar{b}' = B^{-1} b' = \begin{matrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} \end{matrix} \begin{matrix} 3 \\ -3 \end{matrix} = \begin{matrix} 5 \\ -2 \end{matrix}.$$

For a fixed λ, the value of basic variables in table 2 are given by

$$x_1 = \bar{b}_1 + \lambda \bar{b}'_1 = 1 + 5\lambda, \quad x_2 = \bar{b}_2 + \lambda \bar{b}'_2 = 2 - 2\lambda.$$

$\bar{c}_j$ values are not affected as long as the basis consists of variables x_1 and x_2. As λ changes, values of basic variables x_1 and x_2 change and table 2 remains optimal as long as the basis (x_1, x_2) remains feasible. In other words, table 2 remains optimal as long as

$$x_1 = 1 + 5\lambda \geq 0 \text{ or } \lambda \geq -1/5,$$
$$x_2 = 2 - 2\lambda \geq 0 \text{ or } \lambda \leq 1.$$

Therefore, table 2 remains optimal as λ varies from -1/5 to 1. Thus for all $-1/5 \leq \lambda \leq 1$, the optimal solution is given by

$$x_1 = 1 + 5\lambda, \ x_2 = 2 - 2\lambda, \ x_3 = x_4 = x_5 = 0; \ Z_{max} = 16 + 8\lambda.$$

For $\lambda > 1$, the basic variable x_2 becomes negative. Although this makes table 2 infeasible for the primal, it remains feasible for the dual since all $\bar{c}_j$ coefficients are non-positive. Dual simplex method can, therefore, be applied to find the new optimal solution for $\lambda > 1$. Evidently x_2 is the variable that leaves the basis. The ratios of the non basic variables are -3, 10, -2. Thus variable x_4 is the entering variable. the key element -1/3 has been shown bracketed. Regular simplex method is now used to find the new optimal solution. In table 3, the key element has been made unity and x_2 is replaced by x_4.

Table 3

	C_j	4	6	2	0	0		
C_B	**Basis**	X_1	X_2	X_3	X_4	X_5	$\bar{b}$	$\bar{b}'$
4	X_1	**1**	**4**	**7**	**0**	**1**	**9**	**-3**
0	X_4	**0**	**-3**	**-6**	**1**	**-1**	**-6**	**6**
$Z_j = \Sigma c_B a_{ij}$	$\bar{c}_j$	**4**	**16**	**28**	**0**	**4**		
$\bar{c}_j = c_j - Z_j$		**0**	**-10**	**-26**	**0**	**-4**		

The basic solution given by table 3 is

$x_1 = 9 - 3\lambda$, $x_2 = 0$, $x_3 = 0$, $x_4 = -6 + 6\lambda$, $x_5 = 0$, $Z_{max} = 36 - 12\lambda$.

This solution is optimal as long as the basic variables X_1 and X_2 remain non-negative i.e., as long as $x_1 = 9 - 3\lambda \geq 0$ or $\lambda \leq 3$,

$x_2 = -6 + 6\lambda \geq 0$ or $\lambda \geq 1$.

Thus the above solution is optimal for all $1 \leq \lambda \leq 3$.
For $\lambda \leq 3$, the basic variable x_1 becomes negative. As there is no negative coefficient in the first row, the primal solution is infeasible. Hence there exists no optimal solution to the problem for all $\lambda > 3$.

For $\lambda \leq -1/5$, the basic variable x_1 in table 2 becomes negative. Although this makes table 2 infeasible for the primal, it remains feasible for the dual, since all coefficient $\overline{c_j}$ coefficients are non-positive. Dual simplex method can, therefore, be applied to find the new optimal solution for $\lambda \leq -1/5$. Evidently x_1 is the variable that leaves the basis. The ratios of non-basic variables are 6, -5/2, 2. Thus, variable x_5 is the entering variable and -1/3 is the key element. This element is made unity in table 4. Also x_1 is replaced by x_5.

Table 4

	C_j	4	6	2	0	0		
C_B	**Basis**	X_1	X_2	X_3	X_4	X_5	$\bar{b}$	$\bar{b}'$
0	X_5	**-3**	**0**	**3**	**-4**	**1**	**-3**	**-15**
6	X_2	**1**	**1**	**1**	**1**	**0**	**3**	**3**
$Z_j = \Sigma\, c_B\, a_{ij}$		**6**	**6**	**6**	**6**	**0**		
$\overline{c_j} = c_j - Z_j$		**-2**	**0**	**-4**	**-6**	**0**		

The basic solution given by table 4 is

$x_1 = 0$, $x_2 = 3 + 3\lambda$, $x_3 = 0$, $x_4 = 0$, $x_5 = -3 - 15\lambda$, $Z_{max} = 18 + 18\lambda$.

This solution is optimal so long as

$x_2 = 3 + 3\lambda \geq 0$ or $\lambda \geq -1$,

$x_3 = -3 - 15\lambda \geq 0$ or $\lambda \leq -1/5$.

Thus the above solution is optimal for all $-1 \leq \lambda \leq -1/5$.
For λ □ -1, the basic variable x_2 in the table 4 becomes negative. As there is no negative coefficient in the second row, the primal solution is infeasible. Hence there exists no optimal solution to the problem for all λ □ -1. Thus tables 2, 3 and 4 give families of optimal solutions for $-1/5 \leq \lambda \leq 1$, $1 \leq \lambda \leq 3$ and $-1 \leq \lambda \leq -1/5$ respectively.

4.4 EXERCISES

1. Explain parametric linear programming. How does it differ from sensitivity analysis?
2. What are different types of parametric linear programming problems? Explain their solution procedures.

Solve the following

Q1. Consider the parametric problem

maximise $Z = (3-6\lambda)x_1 + (2-2\lambda)x_2 + (5+5\lambda)x_3$

$x_1 + 2x_2 + x_3 \leq 430$

$3x_1 + 2x_3 \leq 460$

$x_1 + 4x_2 \leq 420$

$x_1, x_2, x_3 \geq 0.$

Perform a complete parametric analysis and identify all the critical values of the parameter λ.

Q2. Consider the parametric problem

maximise $Z = 3x_1 + 2x_2 + 5x_3$

$x_1 + 2x_2 + x_3 \leq 430 + \theta$

$3x_1 + 2x_3 \leq 460 - 4\theta$

$x_1 + 4x_2 \leq 420 - 4\theta$

$x_1, x_2, x_3 \geq 0.$

Determine the critical values (range) of θ for which the solution remains optimal basic feasible.

Q3. Q1. Consider the parametric problem

maximise $Z = (3+3\theta)x_1 + 2x_2 + (5-6\theta)x_3$

$x_1 + 2x_2 + x_3 \leq 430$

$3x_1 + 2x_3 \leq 460$

$x_1 + 4x_2 \leq 420$

$x_1, x_2, x_3 \geq 0.$

where θ is a non-negative parameter. Perform a complete parametric analysis.

❖❖❖❖

5

NON-LINEAR PROGRAMMING

Unit Structure

5.1 INTRODUCTION

In linear programming models, the characteristic assumption is the linearity of the objective and constraint functions. Although this assumptions holds in numerous practical situations, yet we come across many situations where the objective function and/or some or all of the constraints are non-linear functions. In some cases, it is possible to formulate a non-linear programming problem into a linear programming model, but generally, specific algorithms are employed for tackling the non-linearity.

A linear programming problem is expressed as

Maximise or minimise $Z = f(x_1, x_2, x_3, ..., x_n)$

Subject to the constraints $g^1 (x_1, x_2, x_3, ..., x_n) \leq, =, \geq b_1$

$g^2 (x_1, x_2, x_3, ..., x_n) \leq, =, \geq b_2$.

. $g^m(x_1, x_2, x_3, ..., x_n) \leq, =, \geq b_{m,}$

$x_j \geq 0, j = 1, 2, 3, ..., n.$

If either the objective function and/or one or more of the constraints are non-linear in X $(x_1, x_2, x_3, ..., x_n)$ the problem is called a non-linear programming problem. In other words, the general non-linear programming problem (NLPP) is to determine the n-tuple $X = (x_1, x_2, x_3, ..., x_n)$ so as to maximise or minimise $Z = f(X)$,

subject to $g^i (X) \leq, =, \geq b_i$, $i = 1, 2, ...,m$.

$X \geq 0$,

where $f(X)$ or some $g^i (X)$ or both are non-linear.

The non-linearity of the functions makes the solution of the problem much more involved as compared to linear programming problems, and there is no single algorithm like the simplex method, which can be employed for this purpose. A number of algorithms have been developed by the researchers, each applicable to a specific type of NLPP only. However, an efficient method for the solution of general non-linear programming problem is still a subject of research.

5.2 ILLUSTRATIVE EXAMPLES

The number of applications of non-linear programming are very large and it is not possible to give a comprehensive survey of all of them. However, some examples illustrating a few of the many situations, where non-linear programming can be applied are given below:

The Product Mix Problem

In the product mix problem, the objective was to determine the product mix, so as to maximise the profits, subject to the constraints on the availability of resources. The objective function was linear as we assumed that there was fixed unit profit associated with each product. This, however, is not always true. In case of large manufacturers, the price of a product is dependent on the quantity demanded. More the volume of sales, lesser the per unit price, which is called advantage of scale. In other words, there is price elasticity. The price-demand curve is not linear, but may look like the one shown in Figure 1, where the price p(x) is very high when x is very small and the price drops rapidly as x increases and then tends to stabilise.

If we assume that unit production and distribution cost of the product is fixed at C, then the profit from x units is given by

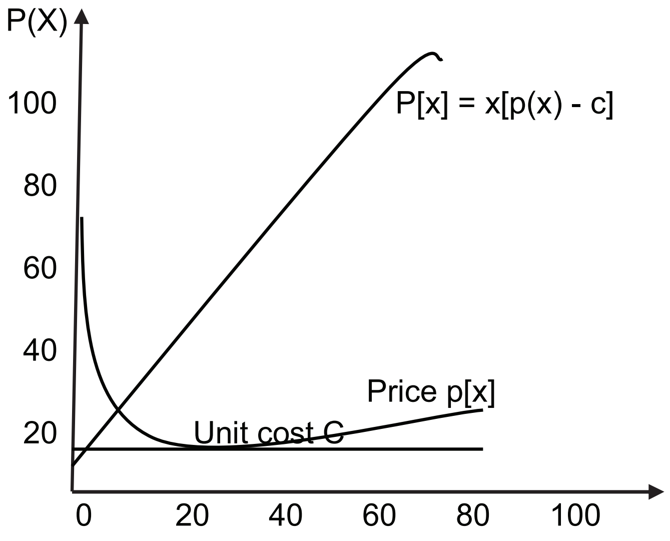
$P(x) = x[p(x) - c] = xp(x) -cx$, which is a non-linear function.

If the manufacturing firm produces n products x_j ($j = 1, 2, ..., n$) which identical profit functions P_j (x_j), then the overall objective function is the sum of n non-linear functions.

$$f(X) = \sum_{j=1}^{n} Pj(xj) \quad ; j = 1, 2, ..., n.$$

In addition to price elasticity, there can be a number of other reasons for the objective function to be non-linear. The unit production cost may decrease with increase in volume of production because of the learning curve effect or it may increase if some special steps are required to be taken to increase the production level.

The constraint functions g^i (X) can also be non-linear, when the use of resources is not strictly proportional to the production levels of respective products.



5.3 TRANSPORTATION PROBLEM

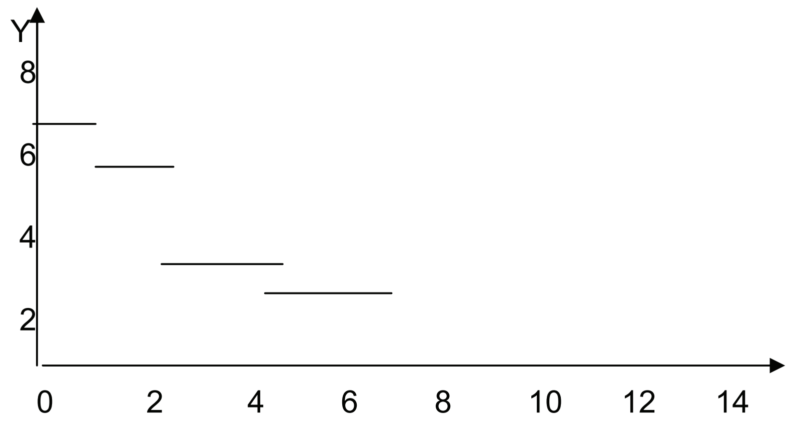
In the transportation problems, it was assumed that the per unit transportation cost from a given source to a given distribution centre was fixed, irrespective of the quantity transported. But in actual practice volume or quantity discounts are generally available. With the increase in volume, the unit transportation cost decreases, as shown in figure 2(i). The resulting total cost C(x) of transporting x units will be shown in figure 2(ii), which is non-linear. This curve is a piece-wise linear function with slope at a point giving the marginal cost at that point. Thus, if each combination of m sources and a destinations has a similar cost function, that is, the cost of transporting x_{ij} units from source i to destination j is given by a non-linear function C_{ij} (X_{ij}), then the overall objective function is

$$\text{maximise } f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij}\,(X_{ij}),$$

where i = 1,, 3, ..., m,

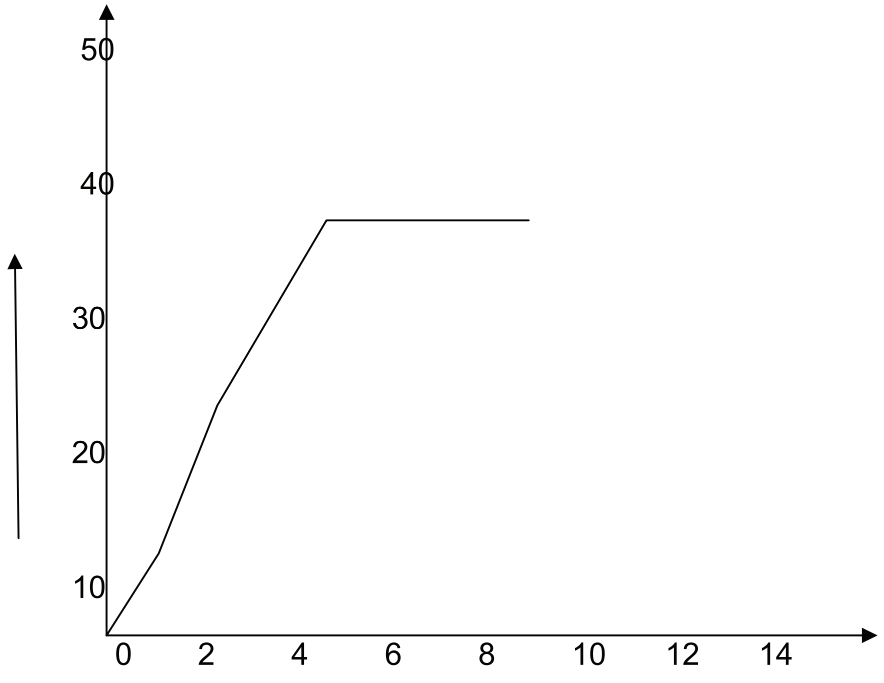
j = 1, 2, 3, ..., n.

The constraints due to the availability at the sources and requirements at the destinations, may remain linear functions.



Quantity Transported

Y = Marginal Cost



Quantity Transported

Y = Total Cost

5.4 PROBLEM FORMULATION EXAMPLES

A manufacturing unit produces two products, the radios and TV sets. The production cost of each product depends upon the number of units being produced. If x_1 and x_2 are the number of radios and TV sets produced, then the production costs are $200\ x_1 + 0.2\ x_2^2$ and $300\ x_2 + 0.2\ x_2^2$ respectively. There is restriction on the production capacity of the radios and TV sets to 100 and 80 units respectively. Similarly, there is restriction on the manpower available. A total 520 man-days are available. The production of one piece of radio requires 2 man-days and one TV set requires 3 man-days.

The sale price is dependent upon the quantity to be produced and the sale relationships are given in table 1.

Product	Quantity Demanded	Unit Price
Radios	2, 000 - 5p	P
TV Sets	4, 000 - 10q	Q

The problem is to determine the number of radios and TV sets which should be produced to maximise the profits.

Formulation of Mathematical Model

Since x_1 and x_2 are the quantities of radios and TV sets to be produced by the firm,

$$x_1 = 2,000 - 5p \text{ or } p = 400 - 0.2\ x_1$$
$$\text{and } x_2 = 4,000 - 10q \text{ 0r } q = 400 - 0.1\ x_2$$

If the total production cost of x_1 unit of radios and x_2 units of TV sets is denoted by C_1 and C_2 respectively, then it is also given that

$$C_1 = 200\ x_1 + 0.2\ x_2^2$$
$$C_2 = 300\ x_2 + 0.2\ x_2^2$$

The total revenue = Revenue of radios + Revenue of TV sets.

i.e., $R = p\ x_1 + q\ x_2$

$= (400 - 0.2\ x_1)\ x_1 + (400 - 0.1\ x_2)\ x_2$

$= 400\ x_1 - 0.2\ x_2^2 + 400\ x_2 - 0.1\ x_2^2.$

 Total profit P is,

$$P = r - (C_1 + C_2)$$

$$= 400\ x_1 - 0.2\ x_1^2 + 400\ x_2 - 0.1\ x_1^2 - 200x_1 - 0.2\ x_1^2 - 300\ x_2 - 0.2\ x_2^2$$

$$= 200\ x_1 - 0.4\ x_1^2 + 100\ x_2 - 0.3\ x_2^2.$$

There are constraints on the production capacity:

$$x_1 \leq 100$$

$$x_2 \leq 80$$

Similarly, man-days available are also limited to 520. Since one unit of radio requires 2 man-days and one unit of TV requires 3 man-days.

$$2\ x_1 + 3\ x_2 \leq 520.$$

Since x_1 and x_2 cannot take negative values,

$$x_1, x_2 \geq 0.$$

Thus the model is

Maximise $f\ (x_1, x_2) = 200\ x_1 - 0.4\ x_1^2 + 100\ x_2 - 0.3\ x_2^2$,

subject to the constraints

$$2\ x_1 + 3\ x_2 \leq 520$$

$$x_1 \leq 100$$

$$x_2 \leq 80$$

$$x_1, x_2 \geq 0.$$

EXAMPLE 2

A company manufactures two products A and B. It takes 30 minutes to process one unit of product A and 15 minutes for one unit of product B. The maximum machine time available is 35 hours per week. One unit of product A requires 2 kg of raw material, while product B requires 3 kg of raw material per unit. The available raw material is limited to 180 kg per week.

The products A and B have unlimited market potential and sell for Rs. 200 and Rs. 500 per unit respectively. If the manufacturing costs for products A and B are $2x^2$ and $3y^2$ respectively, find how much of each product should be produced per week, where x and y are respectively the quantities of A and B to be produced.

Formulation of Mathematical Model

Here x and y are the quantities of products A and B respectively, which are to manufactured per week. The selling price of products A and B is Rs. 200 and Rs. 500 per unit respectively.

 Total revenue per week = $200x + 500y$.

The manufacturing cost of A is $2x^2$ and B is $3y^2$ per unit.
Thus total manufacturing cost per week = $2x^2 + 3y^2$

 Profit per week = $200x + 500y - 2x^2 - 3y^2$.

The machining of product A requires 30 minutes per unit, while product B requires 15 minutes per unit. Since a maximum of 35 hours of machining time are available,

$$30x + 15y \le 35 \times 60$$
$$\text{or } 2x + y \le 140.$$

Constraint on the availability of raw material is expressed as

$$2x + 3y \le 180$$

Since x and y cannot take negative values,

$$x, y \ge 0.$$

Thus the problem can be expressed as,

maximise $\quad f(x, y) = 200x + 500y - 2x^2 - 3y^2$
Subject to $\quad 2x + y \le 140$
$\quad 2x + 3y \le 180$
$\quad x, y \ge 0.$

Here the objective function is non-linear, while constraints are linear.

5.5 TYPES OD NON-LINEAR PROGRAMMING PROBLEMS

There is a wide variety of non-linear programming problems. Some of the most important types are briefly introduced here. Unlike the linear programming, no standard algorithm like the simplex method can be employed to solve the non-linear programming problems. Many different algorithms have been developed to solve the different class of problems.

1. Unconstrained Optimisation

In case of unconstrained optimisation problems, there are no constraints, but only the objective function. The problem is simply to

$$\text{Maximise } f(X)$$

over all values of X = (x_1, x_2, ..., x_n). The necessary condition for a particular solution to be optimal is

$$\frac{\partial f}{\partial xj} = 0 \text{ at } x = X^* \text{ for } j = 1, 2, ..., n.$$

where f(X) is differentiable. This condition is also sufficient conditions when f(X) is a concave function. Thus the solution can be obtained by solving n equations obtained by setting the n partial derivatives equal to zero. However, it is not that simple. For a non-linear function, these equations are often non-linear, and it becomes impossible to solve these equations analytically. Some search procedures have been developed by researchers to solve such problems.

Linearly Constrained Optimisation

As the name indicates, in this type of NLPP, all the constraints g^i (X) are of linear type, while the objective function f (X) is non-linear. Since only one non-linear function is to be handled, the problem becomes comparatively simple to solve. A number of special algorithms have been developed, by extending the simplex method, to handle the non-linear objective function.

Quadratic Programming

This again is a linearly constrained problem, but the objective function f(X) is quadratic. In other words, the objective function contains the terms which are either square of a variable or product of two variables. The general structure of a quadratic programming problem is as follows:

Optimise (Max or Min) z = $\{\sum_{j=1}^{n} cj\ xj + 1/2 \sum_{j=1}^{n} \sum_{k=1}^{n} xj\ djk\ xk$

Subject to $\sum_{j=1}^{n} aij\ xj \leq b_i$, i = 1, 2, 3, ..., n

and $x_j \geq 0$, j = 1, 2, 3, ..., n.

Since, such problems arise very frequently in practical situations, quadratic programming is an important class of NLPP. Many algorithms are available to solve quadratic programming problems.

Convex Programming

Convex programming covers special cases of various types of NLPP, where the objective function f(X) is a concave function and each of the constraints g^i (X) is a convex function. These assumptions ensure that a local maximum is also a global maximum.

Separable Programming

Separable Programming is a special case of convex programming. One additional assumption made here is that all the f(X) and g^i (X) are separable functions.

A function is called separable, when it can be expressed as a sum of functions of individual variables. For example, if f(X) is a separable function, it can be expressed as

$$f(X) = \sum_{i=1}^{n} fi(xi)$$

where each f_i (x_i) is a function comprising of terms involving only x_i.
Consider the two-variable objective function

$$f(x_1, x_2) = 4x_1 + 9x_2 - x_1^2 - x_2^2,$$

which can be expressed as,

$$f(x_1, x_2) = (4x_1 - x_1^2) + (9x_2 - x_2^2)$$
$$= f(x_1) + f(x_2),$$
$$\text{where } f(x_1) = 4x_1 - x_1^2$$
$$\text{and } f(x_2) = 9x_2 - x_2^2$$

The separable programming problems are solved by methods which are extensions of the simplex method.

Non-Convex Programming

All NLPP that do not satisfy the assumptions of convex programming, fall in this category of non-convex programming. There is no algorithm that can result into global optimal solution of such problems. Some algorithms attempt to find the local minima in case the problem does not deviate much from the assumptions of convex programming.

Some specific types of non-convex programming problems can be solved by employing specific methods. Two important types of NLPP under this category are Geometric Programming and Fractional Programming.

5.6 GRAPHICAL METHOD

In linear programming, the solution point is generally a corner point of convex solution space, while in non-linear programming, the solution point is net necessarily a corner or an edge. In other words, the optimal solution is a corner point feasible (CPF) solution.

EXAMPLE 1

Determine the values of x_1 and x_2 so as to

minimise $Z = x_1^2 + x_2^2$

subject to $x_1 + x_2 \geq 8,$

$x_1 + 2x_2 \geq 10,$

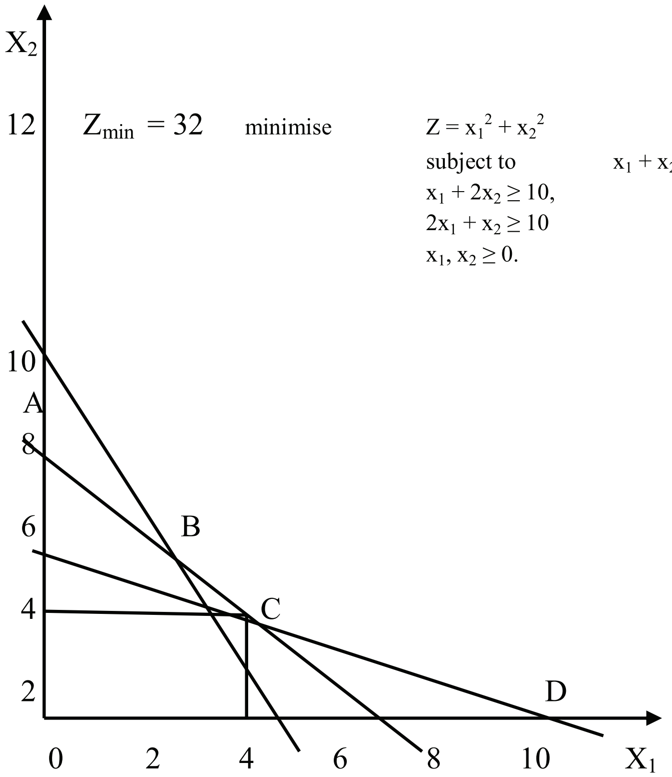
$2x_1 + x_2 \geq 10$

$x_1, x_2 \geq 0.$

Solution

The constraint equations in this case are all linear and give the solution space bounded by a convex region ABCD as shown in the figure 1.

The objective function in this case is non-linear and represents a circle. If r is the radius of the circle, $Z = r^2 - x_1^2 + x_2^{2,}$ then the objective is to determine the minimum value of r, so that the circle with centre (0, 0) and radius r just touches the solution space. In figure 1 desired solution point (4, 4) lies on the line $x_1 + x_2 = 8$, and the line is tangent to the circle at this point.



Note: Since the circle will touch one of the sides of the convex region, one of the sides of the convex solution space would be tangential to the circle and thus the solution can also be obtained as follows:

Differentiate the equation of the circle $x_1^2 + x_2^2 = r^2$

 $2x_1dx_1 + 2x_2dx_2 = 0$

or $\frac{dx2}{dx1} = x_1/ x_2$. .... (1)

Differentiate the constraint equations which form the sides of the convex space.

$dx_1 + dx_2 = 0$ or $\frac{dx2}{dx1} = -1$, ... (2)

$dx_1 + 2dx_2 = 0$ or $\frac{dx2}{dx1} = -1/2$, ...(3)

$2dx_1 + dx_2 = 0$ or $\frac{dx2}{dx1} = -2$. ....(4)

Three solution can be obtained.

Solution 1: Taking equation (1) and (2) and constraints $x_1 + x_2 = 8$,

$\frac{dx2}{dx1} = - x_1/ x_2 = -1$ or $x_1 = x_2$, which gives $x_1 = x_2 = 4$.

This solution satisfies all the constraint equations and is thus feasible.

Solution 2: Taking equation (1) and (3) and constraints $x_1 + x_2 = 10$

$\frac{dx2}{dx1} = - x_1/ x_2 = -1/2$, which gives $x_1 = 2$ and $x_2 = 4$,

This solution does not satisfy the constraints and is discarded.

Solution 3: Taking equation (1) and (4) and constraints **2**$x_1 + x_2 = 10$,

$\frac{dx2}{dx1} = - x_1/ x_2 = -2$ or $x_2 = x_1/2$, which gives $x_1 = 4$ and $x_2 = 2$.

The solution again, does not satisfy the constraints and is discarded. Therefore, optimal solution is $x_1 = 4$; $x_2 = 4$ giving $Z_{min} = 32$.

5.7 QUADRATIC PROGRAMMING

The quadratic programming problem is expressed as

Maximise $f(X) = \sum_{j=1}^{n} cj\ xj + 1/2 \sum_{j=1}^{n} \sum_{k=1}^{n} xj\ djk\ xk$ ....(1)

Subject to $\sum_{j=1}^{n} aij\ xj \leq b_i$,..(2)

and $x_j \geq 0$,..(3)

where i = 1, 2, 3, ..., m and j = 1, 2, 3, ..., n.

KUHN-TRUCKER CONDITIONS

The necessary and sufficient conditions for an optimal solution of quadratic programming problem with objective function of maximisation type and linear constraints, can be obtained as under:

Introduce slack variables s_i^2 and r_j^2 in constraints (2) and (3). The problem modifies to

Maximise $f(X) = \sum_{j=1}^{n} cj\ xj + 1/2 \sum_{j=1}^{n} \sum_{k=1}^{n} xj\ djk\ xk$

subject to $\sum_{j=1}^{n} aij\ xj + s_i^2 = b_i$; i = 1, 2, 3, ..., m

$- x_j + r_j^2 = 0$; j = 1, 2, 3, ..., n.

Now Lagrange function can be written as

$L(X, s, r, \lambda, \mu) = f(X) - \sum_{j=1}^{n} \lambda i\ (a_{ij}x_j + s_i^2 - b_i) - \sum_{j=1}^{n} \mu j\ (-x_j + r_j^2)$

$= \sum_{j=1}^{n} cj\ xj + 1/2 \sum_{j=1}^{n} \sum_{k=1}^{n} xj\ djk\ xk - \sum_{j=1}^{n} \lambda i\ (a_{ij}x_j + s_i^2 - b_i) - \sum_{j=1}^{n} \mu j\ (-x_j + r_j^2)$.

To derive the Kuhn-Trucker necessary conditions, differentiate the Lagrange function partially with respect to its constraint variables and equate each partial derivative to zero.

Differentiating L (X, s, r, λ, μ) with respect to X = $(x_1, x_2, x_3 \ldots, x_n)$

$c_j - \sum_{k=1}^{n} xkdjk - \sum_{i=1}^{m} \lambda i \; a_{ij} + \mu_j = 0$; j = 1, 2, 3, ..., n.. (1)

Differentiating with respect to x

$-2\lambda_s = 0$ or $\lambda_i s_i = 0$ or $\lambda_i s_i^2 = 0$.. (2)

or $\lambda_i \{\sum aij\ xj - b_i\} = 0$

Differentiating w.r.to r.

$-2\mu r = 0$, or $\mu_j r_j = 0$.. (3)

$\mu_j x_j = 0$, j = 1, 2, 3, ..., n.

Differentiating w.r.to λ,

$\sum_{j=1}^{n} aij\ xj + s^2 - b_i = 0$ or $\sum_{j=1}^{n} aij\ xj \leq b_i$, i = 1, 2, 3, ...,m.................(4)

Differentiating with respect to μ,

$-x_j + r_j^2 = 0$

or $x_j \geq 0$, j = 1, 2, 3, ...,n... (5)

And all the variables are non-negative,

$x_j, s_i, r_j, \lambda i, \mu_j \geq 0$.

The conditions number (2) $\lambda_i s_i = 0$ and (3) $\mu_j\ xj = 0$ are called complementary slackness conditions. $\lambda_i s_i = \mu_j\ xj = 0$ implies that the variables xj μ_j and s_i cannot become basic variables simultaneously. Other conditions (1), (4), (5) and (6) are nothing but linear programming constraints in 2(n + m) variables.

5.8 WOLF'S MODIFIED SIMPLEX METHOD

Wolf's method is an extension of the simplex method, where the two phase simplex method has been modified to solve the non-linear programming problem. The method comprises of following iterative steps:

Step 1: Convert the inequality constraints to equations by introducing slack variables s_i^2 and r_i^2 into constraints as discussed earlier.

Step 2: Form the Lagrange function and derive the Kuhn-Trucker conditions.

Step 3: Introduce the artificial variables A_j, j = 1, 2, ..., n in the Kuhn-Trucker conditions and construct an objective function of the type, minimise Z = $\sum_{j=1}^{n} Aj$.

Step 4: Obtain an initial basic feasible solution to the problem

minimise Z = $\sum_{j=1}^{n} Aj$.

Subject to the constraints $\sum_{k=1}^{n} xkdjk$ + $\sum_{i=1}^{m} \lambda i$ $a_{ij} - \mu_j + A_j = C_j$; j = 1, 2, 3, ..., n.

$\sum_{j=1}^{n} aij\ xj$ + $s^2 = b_i$; i = 1, 2, 3, ..., m.

$\lambda_i, x_j, \mu_j, S_i, A_j \geq 0$ for all i and j.

$\lambda_i\ S_i = 0$, Complementary slackness conditions.

$\mu_j\ x_j = 0$ Complementary slackness conditions.

Step 5: Apply two-phase simplex method to obtain the optimal solution to the above problem. This optimal solution is also the optimal solution to the given quadratic programming problem.

EXAMPLE 1:

Solve the following QPP using Wolf's method:

Maximise $Z = 15x_1 + 30x_2 + 4\ x_1\ x_2 - 2\ x_1^2 - 4x_2^2$

Subject to $x_1 + 2\ x_2 \leq 30$,

$x_1, x_2 \geq 0$.

Solution: Consider, non-negativity constraints $x_1, x_2 \geq 0$ as inequality constraints, and add slack variables to all the inequality constraints. The problem becomes

Maximise $Z = 15x_1 + 30x_2 + 4\ x_1\ x_2 - 2\ x_1^2 - 4x_2^2$

Subject to $x_1 + 2\ x_2 + S_1^2 \leq 30$,

$- x_1 + r_1^2 = 0$,

$- x_2 + r_2^2 = 0$

$x_1, x_2, S_1, r_1, r_2 \geq 0$.

Now, construct the Lagrange function
$L(x_1, x_2, S_1, \lambda_1, \mu_1, \mu_2, r_1, r_2) - (15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2) - \lambda_1(x_1 + 2x_2 + S_1^2 - 30) - \mu_1(-x_1 + r_1^2) - \mu_2(-x_2 + r_2^2)$.

The necessary and sufficient conditions for the maximisation of L and hence of Z are:

$\frac{\partial L}{\partial x1} = 15 + 4x_2 - 4x_1 - \lambda_1 + \mu_1 = 0,$

$\frac{\partial L}{\partial x2} = 30 + 4x_1 - 8x_2 - 2\lambda_1 + \mu_2 = 0,$

$\frac{\partial L}{\partial x1} = 2x_2 + x_1 + S_1^2 - 30 = 0,$

$\frac{\partial L}{\partial s1} = 2\lambda_1 S_1 = 0,$

$\frac{\partial L}{\partial \mu 1} = -x_1 + r_1^2 = 0, \frac{\partial L}{\partial \mu 2} = -x_2 + r_2^2 = 0,$

$\frac{\partial L}{\partial r1} = -2\mu_1 r_1 = 0, \frac{\partial L}{\partial r2} = -2\mu_2 r_2 = 0.$

After simplification of these equations, we get

$4x_1 - 4x_2 + \lambda_1 - \mu_1 = 15,$

$-4x_1 + 8x_2 + 2\lambda_1 - \mu_2 = 30,$

$x_1 + 2x_2 + S_1^2 = 30,$

$\left.\begin{array}{l}\lambda_1 S_1 = 0, \\ \mu_1 x_1 = \mu_2 x_2 = 0\end{array}\right\}$ Complementary slackness conditions.

$x_1, x_2, \lambda_1, \mu_1, \mu_2, S_1 \geq 0.$

Now introduce artificial variables A_1 and A_2 in the first two constraint equations and replace S_1^2 in the third constraint equation. The modified LPP becomes
minimise $Z = A_1 + A_2$
subject to the constraints $4x_1 - 4x_2 + \lambda_1 - \mu_1 + A_1 = 15,$
$-4x_1 + 8x_2 + 2\lambda_1 - \mu_2 + A_2 = 30,$
$x_1 + 2x_2 + S_1 = 30,$
$x_1, x_2, \lambda_1, \mu_1, \mu_2, A_1, A2, S1 \geq 0.$

The artificial variables A_1 and A_2 and the slack variable S_1 can be taken as initial basic feasible solution with $x_1 = x_2 = \lambda_1 = \lambda_2 = \mu_1 = \mu_2 = 0$, $A_1 = 15$, $A_2 = 30$ and $S_1 = 30$. The initial basic solution is now put in the tabular form 1.

TABLE 1

	C_j	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**		
C_B	Basis	x_1	X_2	λ_1	μ_1	μ_2	S_1	A_1	A_2	B	θ
1	A_1	4	-4	1	-1	0	0	1	0	15	-
1	A_2	-4	(8)	2	0	-1	0	0	1	30	15/4 →
0	S_1	1	2	0	0	0	1	0	0	30	15/2
$Z_j = \sum CBaij$		0	4	3	-1	-1	0	1	1		
C_j - Zj		0	-4	-3	1	1	0	0	0		
			↑								

From the above table X_2 in the entering variable and A_2 is the leaving variable. Hence replace

A_2 by X_2 in the basis and perform the necessary transformation to obtain table 2.

TABLE 2

	C_j	**0**	**0**	**0**	**0**	**0**	**0**	**1**		
C_B	Basis	x_1	X_2	λ_1	μ_1	μ_2	S_1	A_1	B	θ
1	A_1	2	-0	2	-1	-1/2	0	1	30	15
1	X_2	-1/2	1	1/4	0	-1/8	0	0	30/8	-
0	S_1	(2)	0	-1/2	0	1/4	1	0	45/2	45/4
$Z_j = \sum CBaij$		2	0	2	-1	-1/2	0	1		
C_j - Zj		-2	0	-2	1	1/2	0	0		

Replace the leaving variable S_1 by entering variable x_1 in the current basic solution and write the new table 3.

TABLE 3

	C_j	0	0	0	0	0	0	1		
C_B	Basis	x_1	X_2	λ_1	μ_1	μ_2	S_1	A_1	B	θ →
1	A_1	0	0	(5/2)	-1	-3/4	-1	1	15/2	3
1	X_2	0	1	1/8	0	-3/16	-1/4	0	75/8	75
0	x_1	1	0	-1/4	0	1/8	1/2	0	45/4	-
$Z_j = \sum CBaij$		0	0	5/2	-1	-3/4	-1	1		
C_j - Zj		0	0	-5/2	1	3/4	1	0		
				↑						

Replace A_1 by λ_1 and obtain table 4.

TABLE 4

	C_j	0	0	0	0	0	0	
C_B	Basis	x_1	X_2	λ_1	μ_1	μ_2	S_1	B
0	λ_1	0	0	1	-2/5	-3/10	-2/5	3
0	X_2	0	1	0	1/20	-3/20	-1/5	9
0	x_1	1	0	0	-1/10	1/20	1/20	12
$Z_j = \sum CBaij$		0	0	0	0	0	0	
C_j - Zj		0	0	0	0	0	0	
				↑				

Since all C_j - Zj values are zero, the solution obtained above is optimal.

□ $x_1 = 12$, $X_2 = 9$,

and $Z_{max} = 15\ x_1 + 30\ X_2 + 4\ x_1\ X_2 - 2\ x_1^2 - 4\ x_2^2$
$= 180 + 270 + 432 - 288 - 324$
$= 270$.

5.9 EXERCISES

1. What is meant by Non-Linear Programming.

2. Explain in detail different types of non-linear programming.

Solve the following:

1. Employing graphical method, minimise the distance of the origin from the concave region bounded by the constraints:
$x_1 + x_2 \geq 4$,
$2x_1 + x_2 \geq 5$,
$x_1, x_2 \geq 0$.

2. Determine graphically the values of x_1 and x_2 so as to
maximise $Z = 2x_1 + 3x_2$,
subject to $x_1, x_2 \leq 8$,
$x_1^2 + x_2^2 \leq 20$,
$x_1, x_2 \geq 0$.
Verify that the Kuhn-Tucker conditions hold for the maxima.

3. Use Wolf's method to solve the following quadratic programming problem:
maximise $Z = 2x_1 + x_2 - x_1^2$
Subject to $2x_1 + 3x_2 + S_1^2 \leq 6$,
$2x_1 + 3x_2 \leq 4$
$x_1, x_2 \geq 0$.

❖❖❖❖

6

SOFTWARE APPLICATIONS IN OR

Unit Structure :

6.1 INTRODUCTION

Operations research uses various optimization algorithms to help make decisions related to highly complex problems. Linear Programming (LP) and Mixed Integer Programming (MIP) are often used to solve these highly complex decision-making problems. The operations research procedures available in the NCSS are described below.

6.2 TECHNICAL DETAILS

This page is designed to give a general overview of the capabilities of NCSS for operations research. If you would like to examine the formulas and technical details relating to a specific NCSS procedure, click on the corresponding '[Documentation PDF]' link under each heading to load the complete procedure documentation. There you will find formulas, references, discussions, and examples or tutorials describing the procedure in detail.

6.3 LINEAR PROGRAMMING WITH BOUNDS OR TABLEAU

Linear Programming (LP) maximizes (or minimizes) a linear objective function subject to one or more constraints. The technique finds broad use in operations research and is occasionally of use in statistical work.

The mathematical representation of the linear programming (LP) problem is to maximize (or minimize) the objective function

$$\mathbf{z = CX}$$

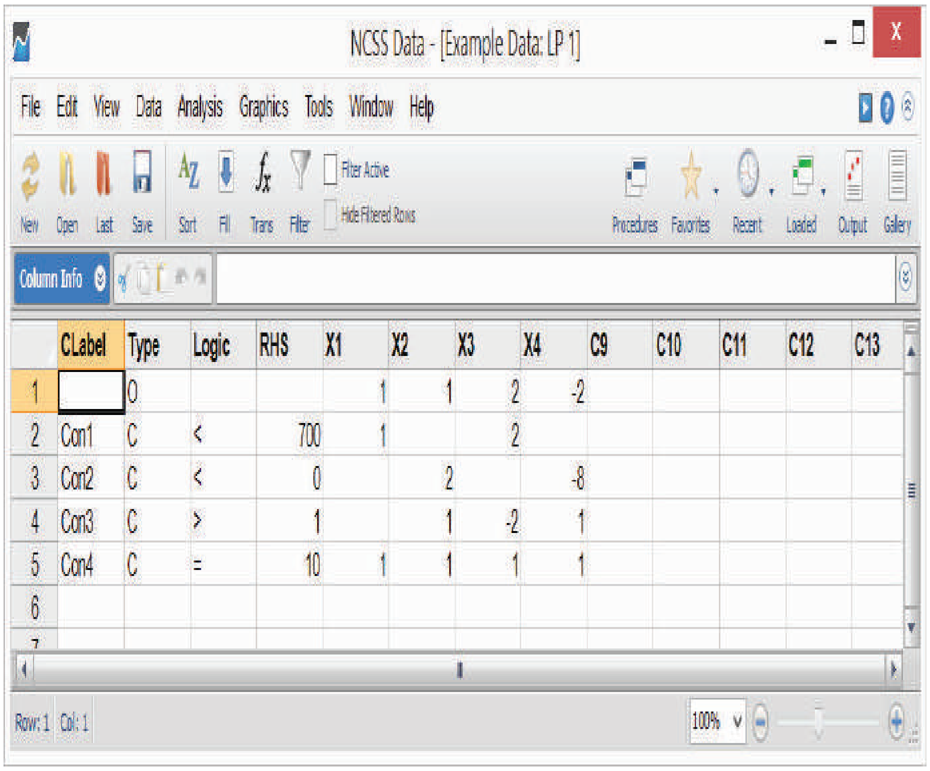
subject to **m** constraints

$$\mathbf{AX \le b, X \ge 0}$$

The values in the **X** vector are called decision variables (the unknowns), and the values in the **b** vector are often called right-hand sides (RHS).

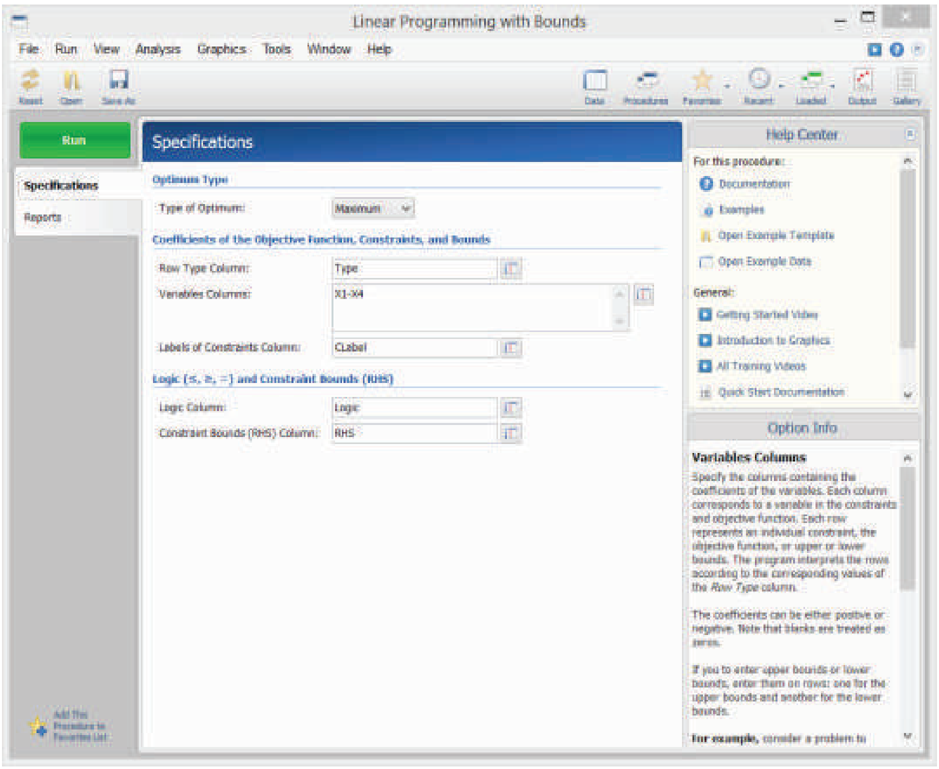
NCSS solves a particular linear program using a revised dual simplex method available in the Extreme Optimization mathematical subroutine package.

SAMPLE DATA


NCSS Data - [Example Data: LP 1]
File Edit View Data Analysis Graphics Tools Window Help
New Open Last Save Sort Fill Trans Filter
Filter Active
Hide Filtered Rows
Procedures Favorites Recent Loaded Output Gallery
Column Info
Row: 1 Col: 1
100%

	CLabel	Type	Logic	RHS	X1	X2	X3	X4	C9	C10	C11	C12	C13
1		O			1	1	2	-2					
2	Con1	C	<	700	1		2						
3	Con2	C	<	0		2		-8					
4	Con3	C	>	1		1	-2	1					
5	Con4	C	=	10	1	1	1	1					
6													

PROCEDURE INPUT



Linear Programming with Bounds

Specifications

Optimum Type

Type of Optimum: Maximum

Coefficients of the Objective Function, Constraints, and Bounds

Row Type Column: Type

Variables Columns: X1-X4

Labels of Constraints Column: CLabel

Logic (≤, ≥, =) and Constraint Bounds (RHS)

Logic Column: Logic

Constraint Bounds (RHS) Column: RHS

Option Info

Variables Columns

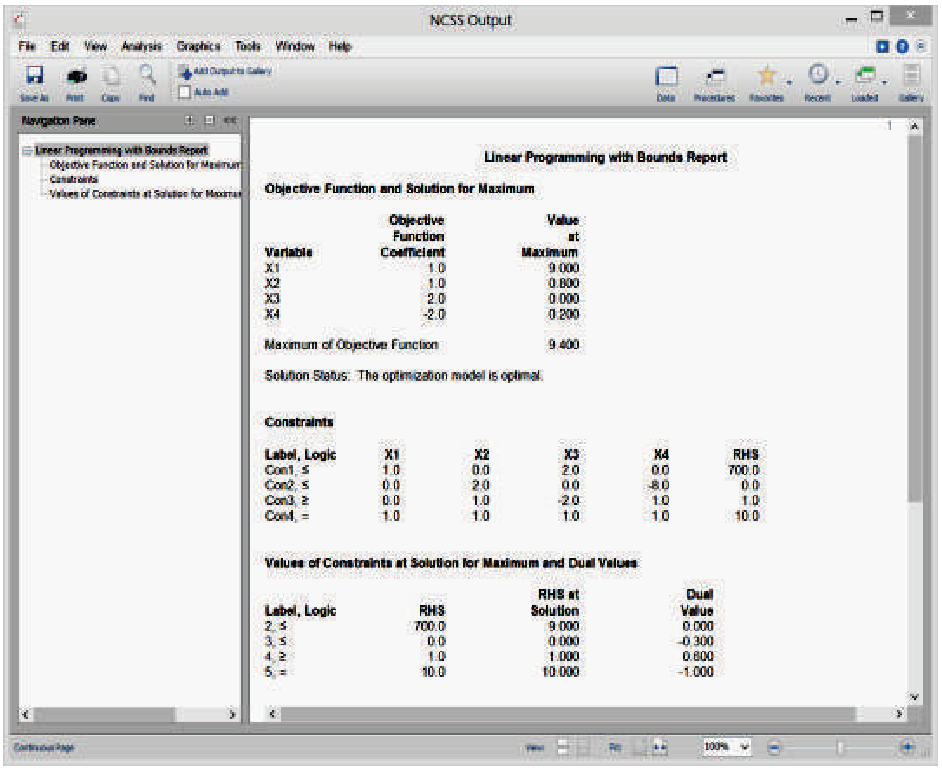
Specify the columns containing the coefficients of the variables. Each column corresponds to a variable in the constraints and objective function. Each row represents an individual constraint, the objective function, or upper or lower bounds. The program interprets the rows according to the corresponding values of the *Row Type* column.

The coefficients can be either positive or negative. Note that blanks are treated as zeros.

If you to enter upper bounds or lower bounds, enter them on rows: one for the upper bounds and another for the lower bounds.

For example, consider a problem to

SAMPLE OUTPUT



NCSS Output

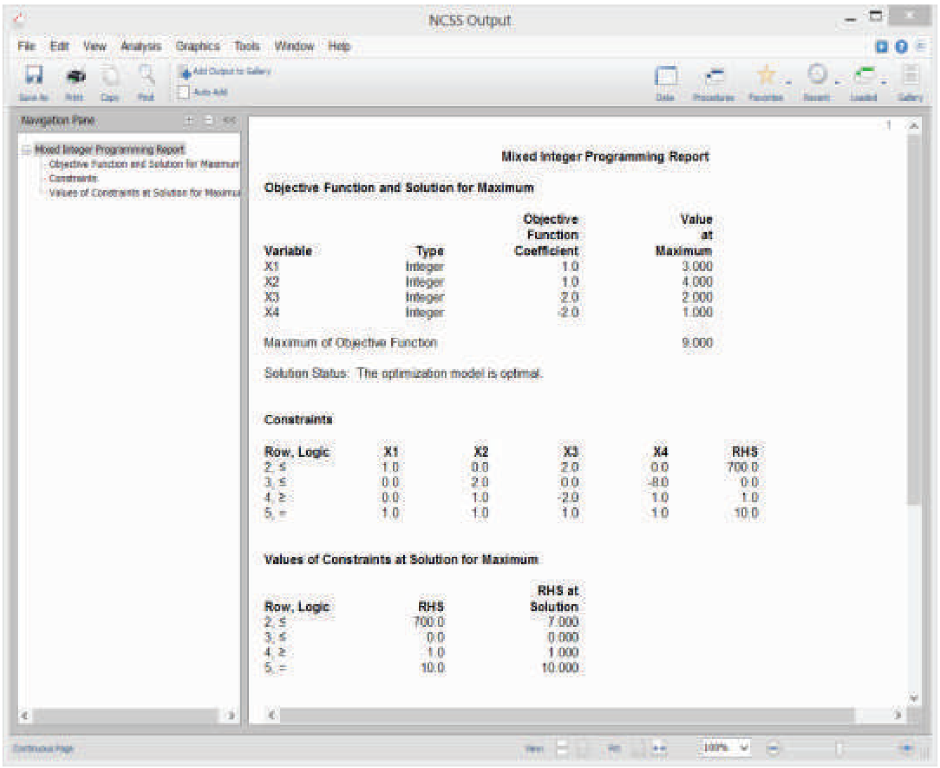
```
                 Linear Programming with Bounds Report

Objective Function and Solution for Maximum

                     Objective               Value
                      Function                  at
Variable           Coefficient             Maximum
X1                         1.0               9.000
X2                         1.0               0.800
X3                         2.0               0.000
X4                        -2.0               0.200

Maximum of Objective Function                9.400

Solution Status:  The optimization model is optimal.


Constraints

Label, Logic      X1          X2          X3          X4          RHS
Con1, ≤          1.0         0.0         2.0         0.0        700.0
Con2, ≤          0.0         2.0         0.0        -8.0          0.0
Con3, ≥          0.0         1.0        -2.0         1.0          1.0
Con4, =          1.0         1.0         1.0         1.0         10.0


Values of Constraints at Solution for Maximum and Dual Values

                                        RHS at                Dual
Label, Logic               RHS        Solution               Value
2, ≤                     700.0           9.000               0.000
3, ≤                       0.0           0.000              -0.300
4, ≥                       1.0           1.000               0.800
5, =                      10.0          10.000              -1.000
```

MIXED INTEGER PROGRAMMING

Linear programming maximizes (or minimizes) a linear objective function subject to one or more constraints. Mixed Integer Programming (MIP) adds one additional condition that at least one of the variables can only take on integer values. The technique finds broad use in operations research.

NCSS solves a particular mixed integer programming problem using the branch and bound algorithm available in the Extreme Optimization mathematical subroutine package.

SAMPLE OUTPUT


NCSS Output
File Edit View Analysis Graphics Tools Window Help
Navigation Pane
Mixed Integer Programming Report
Objective Function and Solution for Maximum
Constraints
Values of Constraints at Solution for Maximum

```
                    Mixed Integer Programming Report

Objective Function and Solution for Maximum

                                   Objective            Value
                                   Function                at
Variable            Type           Coefficient        Maximum
X1                  Integer                1.0          3.000
X2                  Integer                1.0          4.000
X3                  Integer                2.0          2.000
X4                  Integer               -2.0          1.000

Maximum of Objective Function                           9.000

Solution Status:  The optimization model is optimal.


Constraints

Row, Logic       X1          X2          X3          X4          RHS
2, ≤            1.0         0.0         2.0         0.0        700.0
3, ≤            0.0         2.0         0.0        -8.0          0.0
4, ≥            0.0         1.0        -2.0         1.0          1.0
5, =            1.0         1.0         1.0         1.0         10.0


Values of Constraints at Solution for Maximum

                                    RHS at
Row, Logic          RHS           Solution
2, ≤              700.0              7.000
3, ≤                0.0              0.000
4, ≥                1.0              1.000
5, =               10.0             10.000
```

QUADRATIC PROGRAMMING

Quadratic Programming maximizes (or minimizes) a quadratic objective function subject to one or more constraints. The technique finds broad use in operations research and is occasionally of use in statistical work.

The mathematical representation of the quadratic programming (QP) problem is to maximize the objective function

$$z = CX + (1/2)X'HX \text{ or } z = CX + X'DX$$

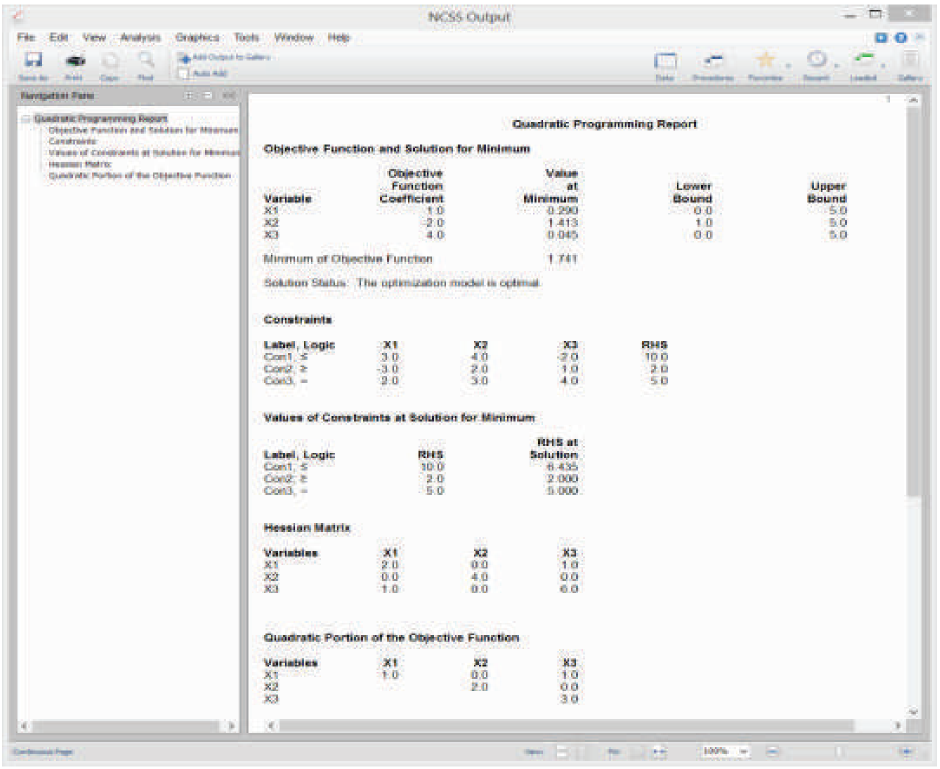
subject to **m** constraints

$$AX \leq b, X \geq 0$$

The values in the **X** vector are called decision variables (the unknowns), and the values in the **b** vector are often called right-hand sides (RHS).

NCSS solves a particular quadratic program using a primal active set method available in the Extreme Optimization mathematical subroutine package.

SAMPLE OUTPUT

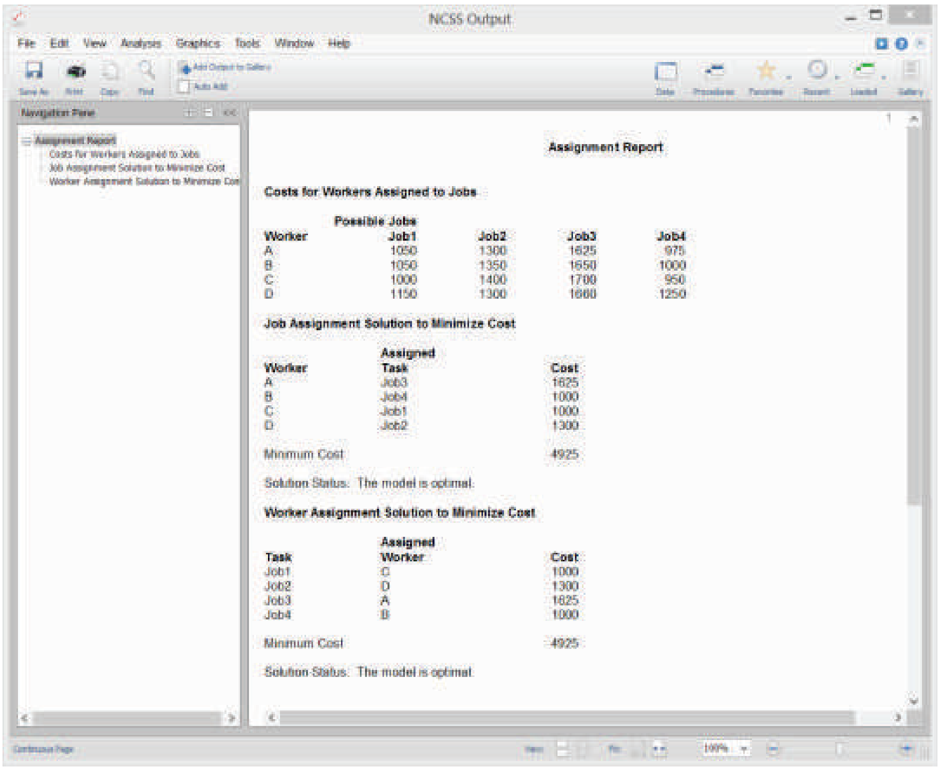


```
                         Quadratic Programming Report

Objective Function and Solution for Minimum

                    Objective           Value
                     Function              at          Lower          Upper
Variable          Coefficient         Minimum          Bound          Bound
X1                        1.0           0.290            0.0            5.0
X2                       -2.0           1.413            1.0            5.0
X3                        4.0           0.045            0.0            5.0

Minimum of Objective Function           1.741

Solution Status:  The optimization model is optimal.


Constraints

Label, Logic        X1          X2          X3          RHS
Con1, ≤            3.0         4.0        -2.0         10.0
Con2, ≥           -3.0         2.0         1.0          2.0
Con3, =            2.0         3.0         4.0          5.0

Values of Constraints at Solution for Minimum

                                            RHS at
Label, Logic              RHS             Solution
Con1, ≤                  10.0                6.435
Con2, ≥                   2.0                2.000
Con3, =                   5.0                5.000

Hessian Matrix

Variables           X1          X2          X3
X1                 2.0         0.0         1.0
X2                 0.0         4.0         0.0
X3                 1.0         0.0         6.0


Quadratic Portion of the Objective Function

Variables           X1          X2          X3
X1                 1.0         0.0         1.0
X2                             2.0         0.0
X3                                         3.0
```

ASSIGNMENT

The object of the Assignment algorithm is to assign **n** objects (workers, machines, etc.) to the same number of jobs (tasks) in such a way that will minimize the total cost. The problem assumes that only one task is assigned to each object. NCSS solves the problem using the mixed integer programming algorithm available in the Extreme Optimization mathematical subroutine package.

SAMPLE OUTPUT

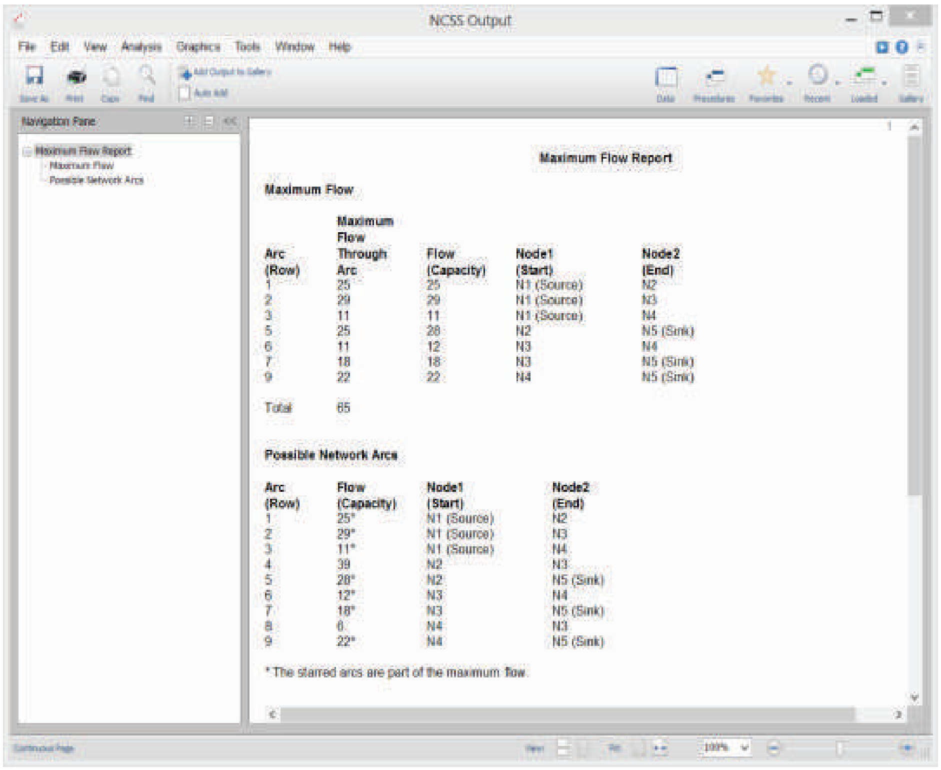

NCSS Output
File Edit View Analysis Graphics Tools Window Help
Navigation Pane
Assignment Report
Costs for Workers Assigned to Jobs
Job Assignment Solution to Minimize Cost
Worker Assignment Solution to Minimize Cost

```
                              Assignment Report

Costs for Workers Assigned to Jobs

              Possible Jobs
Worker             Job1          Job2          Job3          Job4
A                  1050          1300          1625           975
B                  1050          1350          1650          1000
C                  1000          1400          1700           950
D                  1150          1300          1660          1250

Job Assignment Solution to Minimize Cost

                   Assigned
Worker             Task                        Cost
A                  Job3                        1625
B                  Job4                        1000
C                  Job1                        1000
D                  Job2                        1300

Minimum Cost                                   4925

Solution Status:  The model is optimal.

Worker Assignment Solution to Minimize Cost

                   Assigned
Task               Worker                      Cost
Job1               C                           1000
Job2               D                           1300
Job3               A                           1625
Job4               B                           1000

Minimum Cost                                   4925

Solution Status:  The model is optimal.
```

MAXIMUM FLOW

Given a directed network defined by nodes, arcs, and flow capacities, this procedure finds the maximum flow that can occur between a source node and a sink node. An example of this is the flow of oil through a pipeline with several junctions. NCSS uses the linear programming approach to solve the problem as outlined in Taha (2011) and Hillier and Lieberman (2015).

SAMPLE OUTPUT

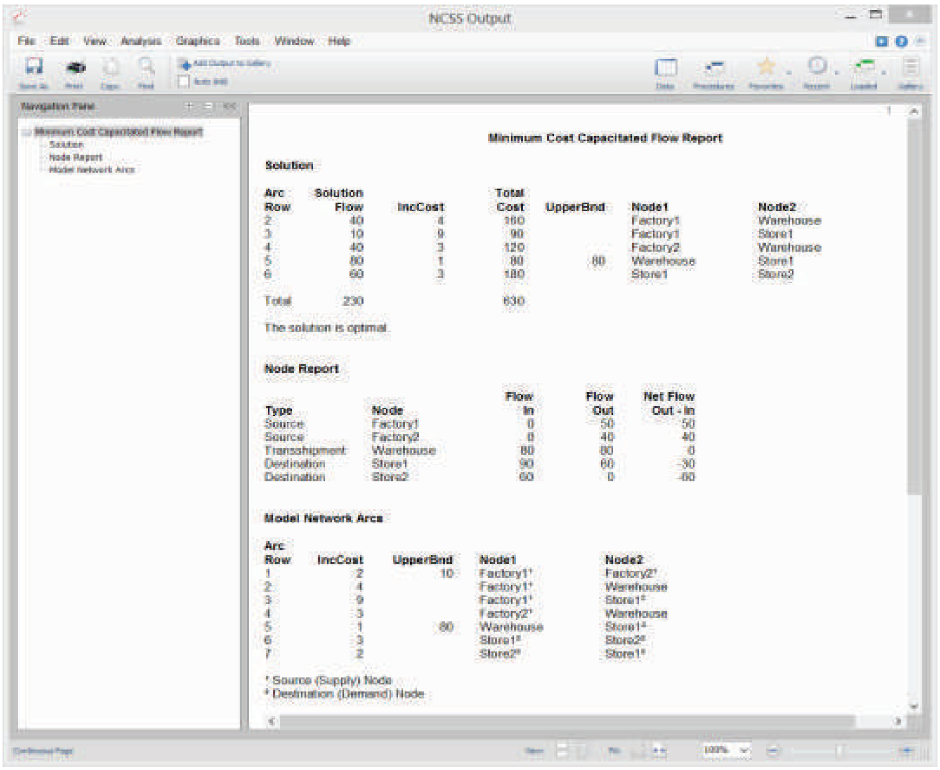


```
                         Maximum Flow Report

Maximum Flow

        Maximum
        Flow
Arc     Through    Flow        Node1          Node2
(Row)   Arc        (Capacity)  (Start)        (End)
1       25         25          N1 (Source)    N2
2       29         29          N1 (Source)    N3
3       11         11          N1 (Source)    N4
5       25         28          N2             N5 (Sink)
6       11         12          N3             N4
7       18         18          N3             N5 (Sink)
9       22         22          N4             N5 (Sink)

Total   65

Possible Network Arcs

Arc     Flow        Node1          Node2
(Row)   (Capacity)  (Start)        (End)
1       25*         N1 (Source)    N2
2       29*         N1 (Source)    N3
3       11*         N1 (Source)    N4
4       39          N2             N3
5       28*         N2             N5 (Sink)
6       12*         N3             N4
7       18*         N3             N5 (Sink)
8       6           N4             N3
9       22*         N4             N5 (Sink)

* The starred arcs are part of the maximum flow.
```

MINIMUM COST CAPACITATED FLOW

The Minimum Cost Capacitated Flow model is prominent among network flow models because so many other network models are special cases. The maximum flow, shortest-path, transportation, transhipment, and assignment models are all special cases of this model. NCSS uses the linear programming approach to solve the problem as outlined in Hillier and Lieberman (2015).

SAMPLE OUTPUT



NCSS Output
File Edit View Analysis Graphics Tools Window Help
Navigation Pane
Minimum Cost Capacitated Flow Report
Solution
Node Report
Model Network Arcs

```
                    Minimum Cost Capacitated Flow Report

Solution

Arc    Solution                  Total
Row        Flow     IncCost       Cost    UpperBnd    Node1          Node2
2            40           4        160                Factory1       Warehouse
3            10           9         90                Factory1       Store1
4            40           3        120                Factory2       Warehouse
5            80           1         80          80    Warehouse      Store1
6            60           3        180                Store1         Store2

Total       230                    630

The solution is optimal.

Node Report

                                       Flow       Flow    Net Flow
Type           Node                      In        Out    Out - In
Source         Factory1                   0         50          50
Source         Factory2                   0         40          40
Transshipment  Warehouse                 80         80           0
Destination    Store1                    90         60         -30
Destination    Store2                    60          0         -60

Model Network Arcs

Arc
Row     IncCost     UpperBnd    Node1          Node2
1             2           10    Factory1¹      Factory2¹
2             4                 Factory1¹      Warehouse
3             9                 Factory1¹      Store1²
4             3                 Factory2¹      Warehouse
5             1           80    Warehouse      Store1²
6             3                 Store1²        Store2²
7             2                 Store2²        Store1²

¹ Source (Supply) Node
² Destination (Demand) Node
```

MINIMUM SPANNING TREE

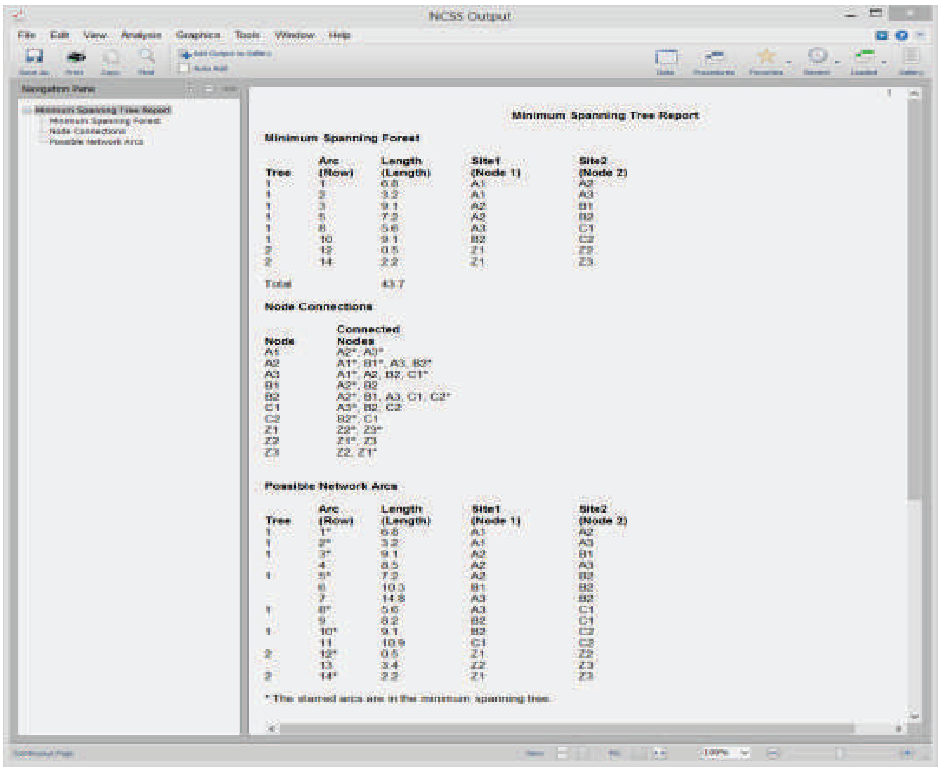
A Minimum Spanning Tree links all nodes (points or vertices) of a network with the minimum length among all the arcs. This procedure finds the minimum spanning tree of a network using a greedy algorithm. If the network is not connected, the solution, called a minimum spanning forest, is a combination of minimum spanning trees formed on the connected subsets.

The algorithm can be used in applications such as transportation networks, computer networks, and water networks where a tree connecting all nodes must have minimum length.

The algorithm proceeds as follows:

- 1. Start with any node.
- 2. Connect this node to its nearest neighbor using any of the available branches.
- 3. Find the unconnected node that is closest any of the connected nodes. Connect these nodes.
- 4. Repeat steps 2 and 3 until all nodes have been connected.

SAMPLE OUTPUT

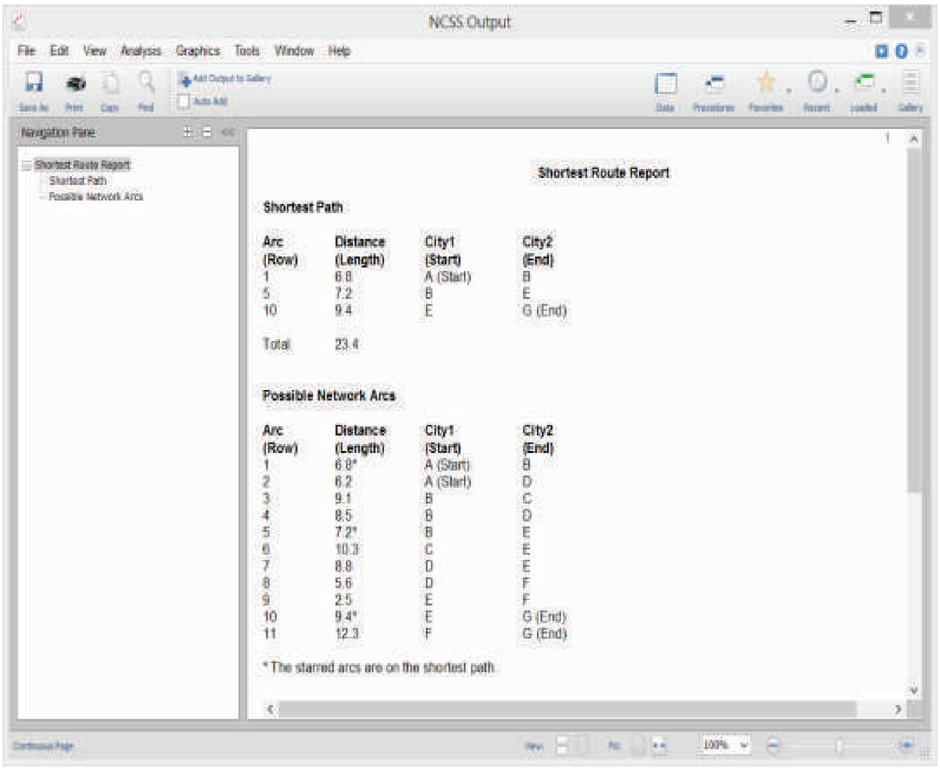


```
Minimum Spanning Tree Report

Minimum Spanning Forest

          Arc        Length       Site1          Site2
Tree      (Row)      (Length)     (Node 1)       (Node 2)
1         1          6.8          A1             A2
1         2          3.2          A1             A3
1         3          9.1          A2             B1
1         5          7.2          A2             B2
1         8          5.6          A3             C1
1         10         9.1          B2             C2
2         12         0.5          Z1             Z2
2         14         2.2          Z1             Z3

Total                43.7

Node Connections

             Connected
Node         Nodes
A1           A2*, A3*
A2           A1*, B1*, A3, B2*
A3           A1*, A2, B2, C1*
B1           A2*, B2
B2           A2*, B1, A3, C1, C2*
C1           A3*, B2, C2
C2           B2*, C1
Z1           Z2*, Z3*
Z2           Z1*, Z3
Z3           Z2, Z1*

Possible Network Arcs

          Arc        Length       Site1          Site2
Tree      (Row)      (Length)     (Node 1)       (Node 2)
1         1*         6.8          A1             A2
1         2*         3.2          A1             A3
1         3*         9.1          A2             B1
          4          8.5          A2             A3
1         5*         7.2          A2             B2
          6          10.3         B1             B2
          7          14.8         A3             B2
1         8*         5.6          A3             C1
          9          8.2          B2             C1
1         10*        9.1          B2             C2
          11         10.9         C1             C2
2         12*        0.5          Z1             Z2
          13         3.4          Z2             Z3
2         14*        2.2          Z1             Z3

* The starred arcs are in the minimum spanning tree.
```

SHORTEST ROUTE

Given a directed network defined by nodes and arcs, this procedure finds the shortest route between two specified nodes. One example of the need for such an algorithm is to be used in a GPS device to find the shortest route between two locations. NCSS uses the linear programming approach to solve the problem as outlined in Taha (2011).

SAMPLE OUTPUT

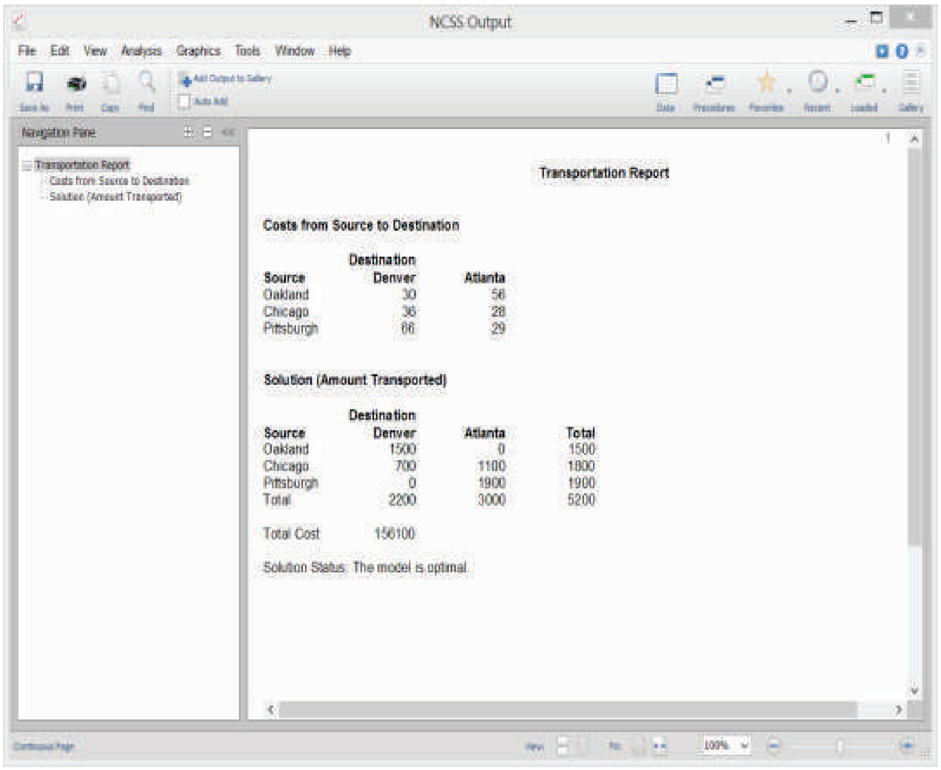


```
                    Shortest Route Report

Shortest Path

Arc      Distance   City1       City2
(Row)    (Length)   (Start)     (End)
1        6.8        A (Start)   B
5        7.2        B           E
10       9.4        E           G (End)

Total    23.4

Possible Network Arcs

Arc      Distance   City1       City2
(Row)    (Length)   (Start)     (End)
1        6.8*       A (Start)   B
2        6.2        A (Start)   D
3        9.1        B           C
4        8.5        B           D
5        7.2*       B           E
6        10.3       C           E
7        8.8        D           E
8        5.6        D           F
9        2.5        E           F
10       9.4*       E           G (End)
11       12.3       F           G (End)

* The starred arcs are on the shortest path.
```

TRANSPORTATION

The object of the Transportation algorithm is to find the amounts shipped from m sources to n destinations that will minimize the total cost of distribution while meeting the demands at each destination and staying within the amount that can be supplied from each source. The problem assumes that only whole units can be shipped. NCSS solves the problem using the Mixed Integer Programming algorithm available in the Extreme Optimization mathematical subroutine package.

SAMPLE OUTPUT

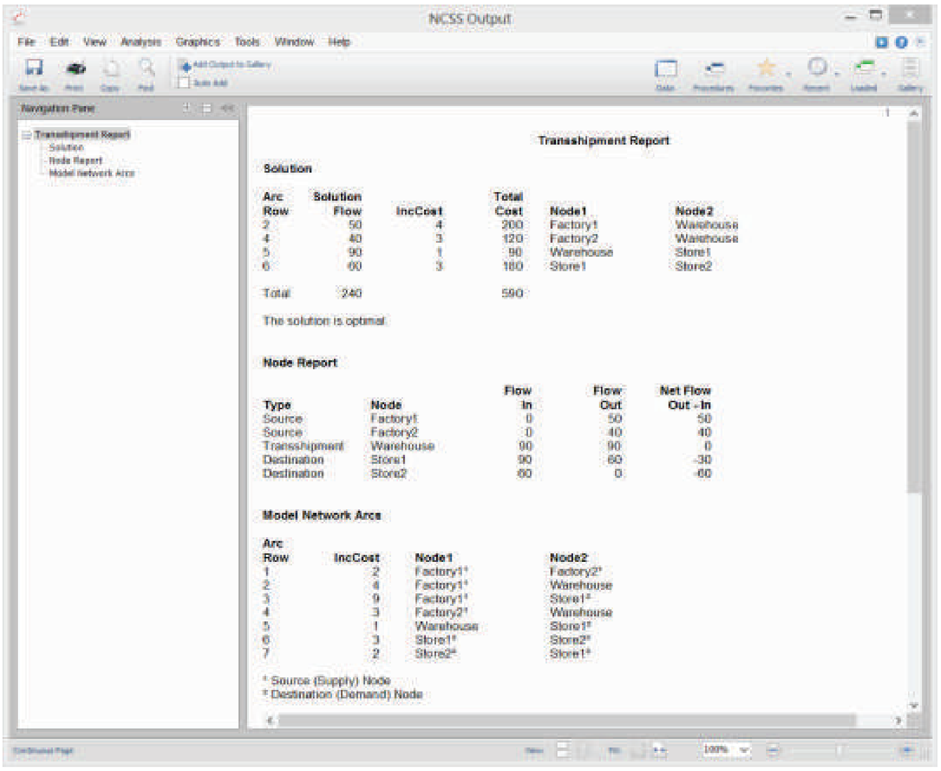

NCSS Output
File Edit View Analysis Graphics Tools Window Help
Navigation Pane
Transportation Report
Costs from Source to Destination
Solution (Amount Transported)

```
                    Transportation Report

Costs from Source to Destination

              Destination
Source        Denver       Atlanta
Oakland           30            56
Chicago           36            28
Pittsburgh        66            29

Solution (Amount Transported)

              Destination
Source        Denver       Atlanta       Total
Oakland         1500             0        1500
Chicago          700          1100        1800
Pittsburgh         0          1900        1900
Total           2200          3000        5200

Total Cost    156100

Solution Status: The model is optimal.
```

TRANSSHIPMENT

The Transshipment model is a special case of the minimum cost capacitated flow model in which there are no capacities or minimums on the arc flows. The transshipment model is similar to a transportation model, except that it allows the more realistic assumption that all nodes can transfer to and from all other nodes, no matter what their node type. Hence, it allows product to be shipped between sources and between destinations, an ability that is missing in the transportation model. NCSS uses the linear programming approach to solve the problem as outlined in Hillier and Lieberman (2015).

SAMPLE OUTPUT



```
                              Transhipment Report

Solution

Arc       Solution                   Total
Row           Flow      IncCost       Cost    Node1          Node2
2               50            4        200    Factory1       Warehouse
4               40            3        120    Factory2       Warehouse
5               90            1         90    Warehouse      Store1
6               60            3        180    Store1         Store2

Total          240                     590

The solution is optimal.

Node Report

                                  Flow        Flow      Net Flow
Type            Node                In         Out      Out - In
Source          Factory1             0          50            50
Source          Factory2             0          40            40
Transshipment   Warehouse           90          90             0
Destination     Store1              90          60           -30
Destination     Store2              60           0           -60

Model Network Arcs

Arc
Row        IncCost    Node1           Node2
1                2    Factory1[S]     Factory2[S]
2                4    Factory1[S]     Warehouse
3                9    Factory1[S]     Store1[D]
4                3    Factory2[S]     Warehouse
5                1    Warehouse       Store1[D]
6                3    Store1[D]       Store2[D]
7                2    Store2[D]       Store1[D]

[S] Source (Supply) Node
[D] Destination (Demand) Node
```

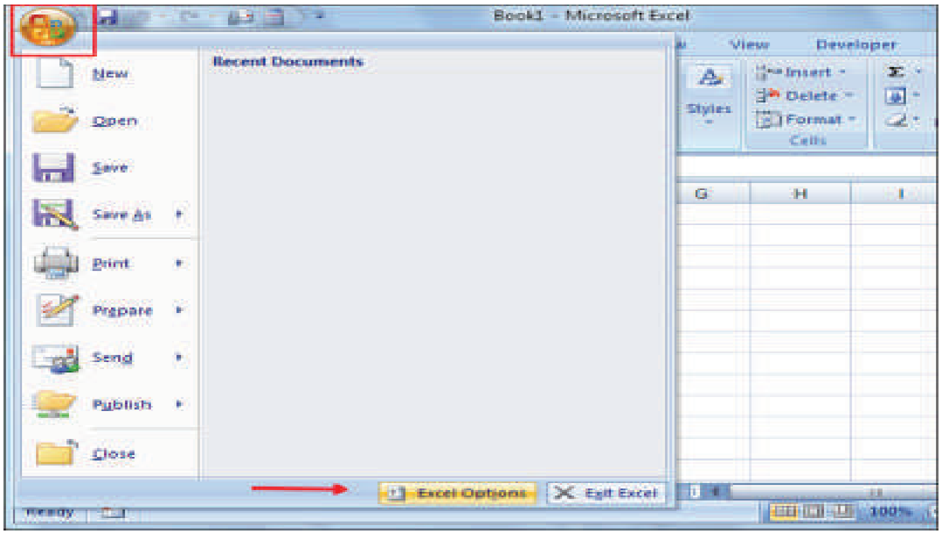
6.4 HOW TO SOLVE EQUATIONS IN EXCEL USING SOLVER ADD-IN

Microsoft Excel is a great Office application from Microsoft and it does not need any introduction. It helps every one of us, in many ways by making our tasks simpler. In this post we will see how to **solve Equations in Excel**, using Solver Add-in.

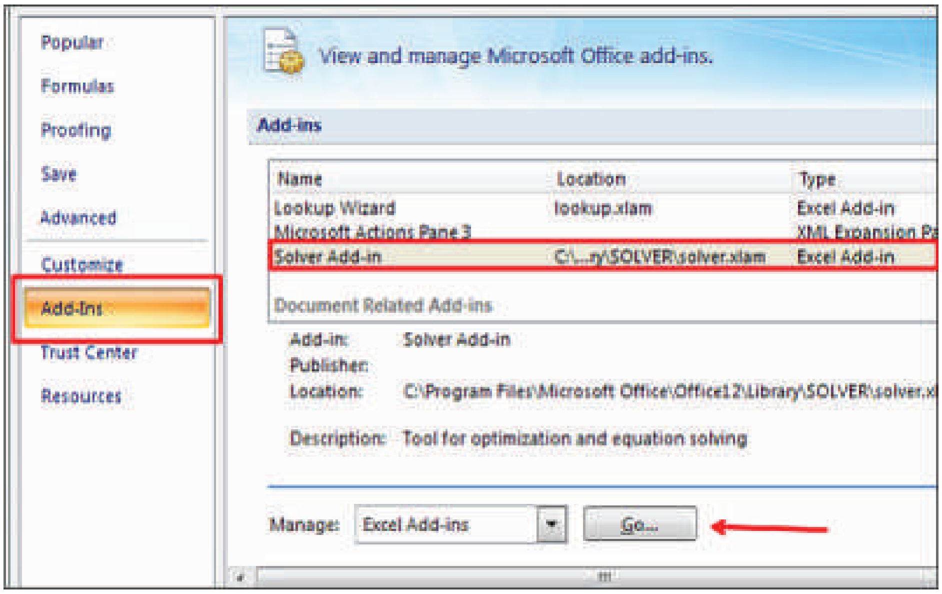
Some or the other day, you might have come across the need to carry out reverse calculations. For example, you might need to calculate values of two variables which satisfy the given two equations. You will try to figure out the values of variables satisfying the equations. Another example would be – the exact marks needed in the last semester to complete your graduation. So, we have total marks needed to complete the graduation and the sum of all marks of the previous semesters. We use these inputs and perform some mathematical calculations to figure out the exact marks needed in the last semester. This entire process and calculations can be simple and easily made with the help of Excel using **Solver Add-in.**

SOLVE EQUATIONS IN EXCEL

Solver Add-in powerful and useful tool of Excel which performs calculations to give the optimal solutions meeting the specified criteria. So, let us see how to use Solver Add-in for Excel. Solver Add-in is not loaded in to excel by default and we need to load it as follows, Open Excel and click on File or Office Button, then click on **Excel Options.**



Excel Options dialog box opens up and click on **Add-ins** on the left side. Then, select **Solver Add-in** from the list and Click on "**Go**" button.



View and manage Microsoft Office add-ins.

Add-ins

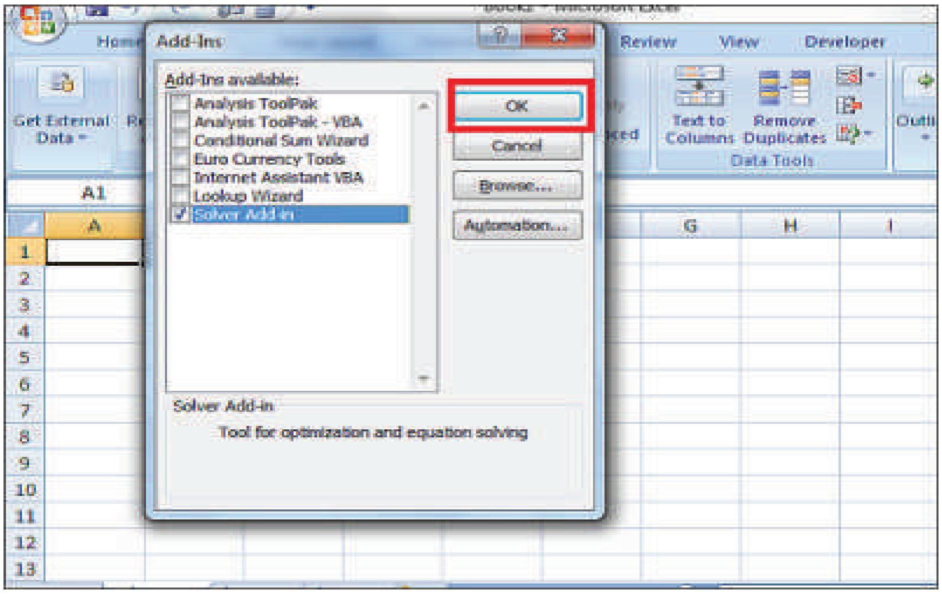
Name	Location	Type
Lookup Wizard	lookup.xlam	Excel Add-in
Microsoft Actions Pane 3		XML Expansion Pa
Solver Add-in	C:\...ry\SOLVER\solver.xlam	Excel Add-in

Document Related Add-ins

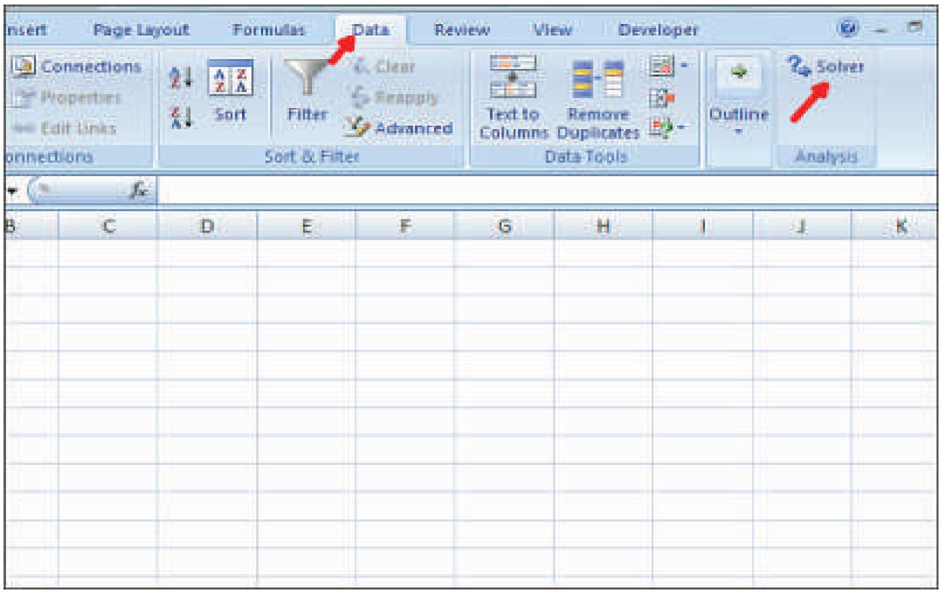
Add-in: Solver Add-in
Publisher:
Location: C:\Program Files\Microsoft Office\Office12\Library\SOLVER\solver.x
Description: Tool for optimization and equation solving

Manage: Excel Add-ins

Add-ins dialog box shows list of add-ins. Select the Solver Add-in and click "Ok" button.

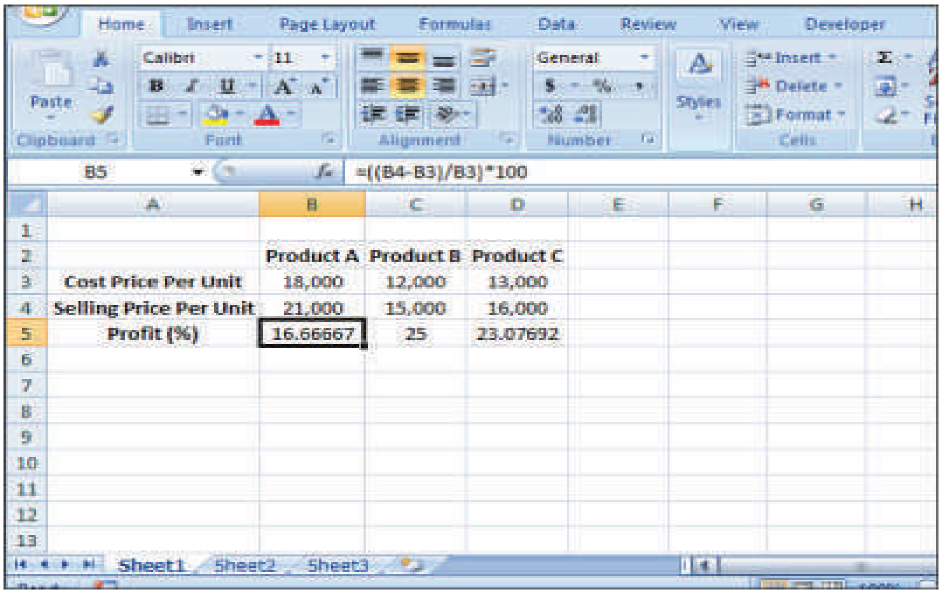


Now, Solver Add-in got added to the Excel sheet. Tap on the "Data" tab and on the extreme right, you can see the added Solver Add-in.



HOW TO USE SOLVER ADD-IN

We added Solver Add-in to Excel and now we will see how to use it. To understand it better, let us take an example of calculating the profit of a product. See the Excel sheet below with some sample data in it. To find the profit %, we use the formula **profit %=((Selling price-Cost price)/Cost price)*100**



B5 =((B4-B3)/B3)*100

	A	B	C	D
1				
2		Product A	Product B	Product C
3	Cost Price Per Unit	18,000	12,000	13,000
4	Selling Price Per Unit	21,000	15,000	16,000
5	Profit (%)	16.66667	25	23.07692

We can see that there are three products as Product A, Product B and Product C with Cost Price, Selling Price and Profit (%) of respective products. Now, our target is to take the profit (%) of Product A to 20%. We need to find out the Cost Price and Selling Price values of Product A needed to make the profit as 20%. Here, we also have the constraint that Cost Price should be greater than or equal to 16,000 and Selling Price should be less than or equal top 22,000. So, first we need to list down the below information based on the example we took.

Target Cell: B5 (Profit %)

Variable Cells for Product A: B3 (Cost Price) and B4 (Selling Price)

Constraints: B3 >= 16,000 and B4 <= 22,000

Formula used to calculate profit %: ((Selling price-Cost price)/Cost price)*100

Target Value: 20

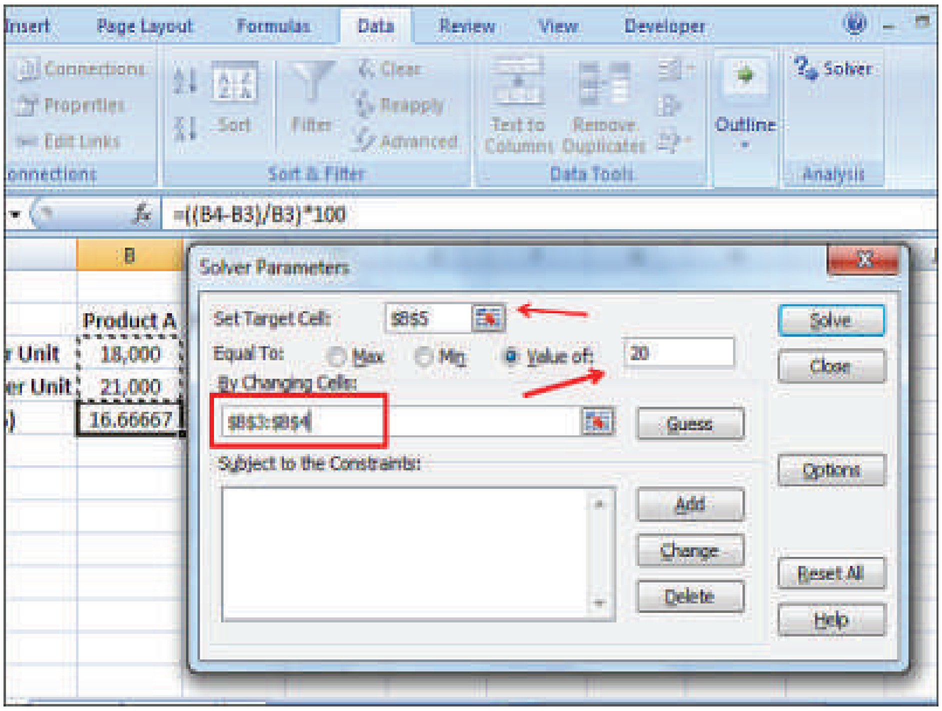
Place the formula in the target cell (**B5)** to calculate the profit %.

This is the required information we need to solve any sort of equation using Solver Add-in in Excel.

Now, launch the Solver Add-in by clicking on the Data tab and click on Solver.

STEP 1: Specify the “Target Cell” as **B5**, “Value of” as the targeted profit % as 20 and specify the cells which need to be changed to meet the required profit %. In our case, **B3 (C.P)** and **B4**

(S.P) need to be specified as **$B$3:$B$4** in "By changing variable cells".



=((B4-B3)/B3)*100

	B
	Product A
Unit	18,000
er Unit	21,000
)	16.66667

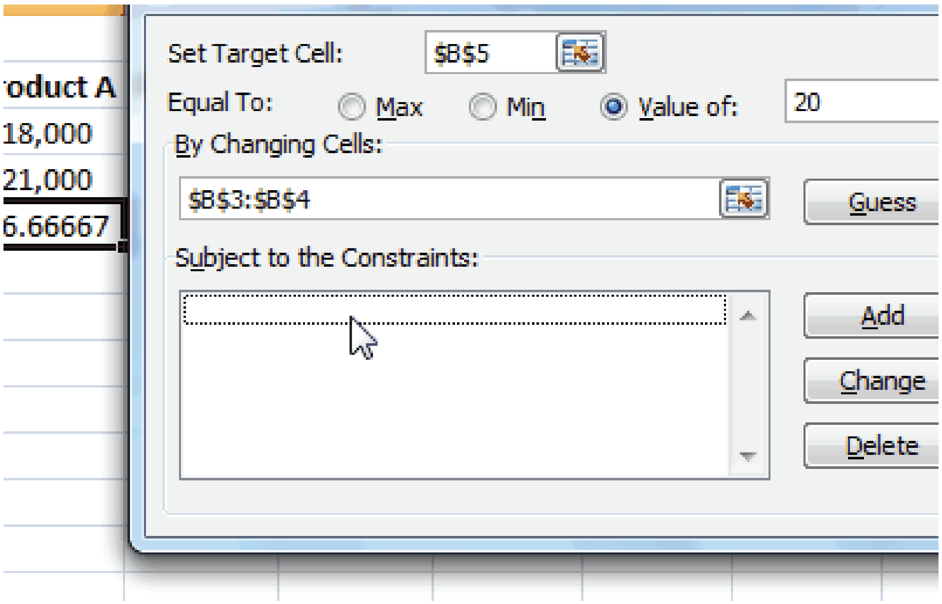
Set Target Cell: $B$5

Equal To: ○ Max ○ Min [● Value of:] 20

By Changing Cells: $B$3:$B$4

Subject to the Constraints:

STEP 2: Now, it's time to add constraints. In our case, Cost Price (B3) >=16,000 and Selling Price (B4) <=22,000. Click on the "Add" button and add constraints as follows.



oduct A
18,000
21,000
6.66667

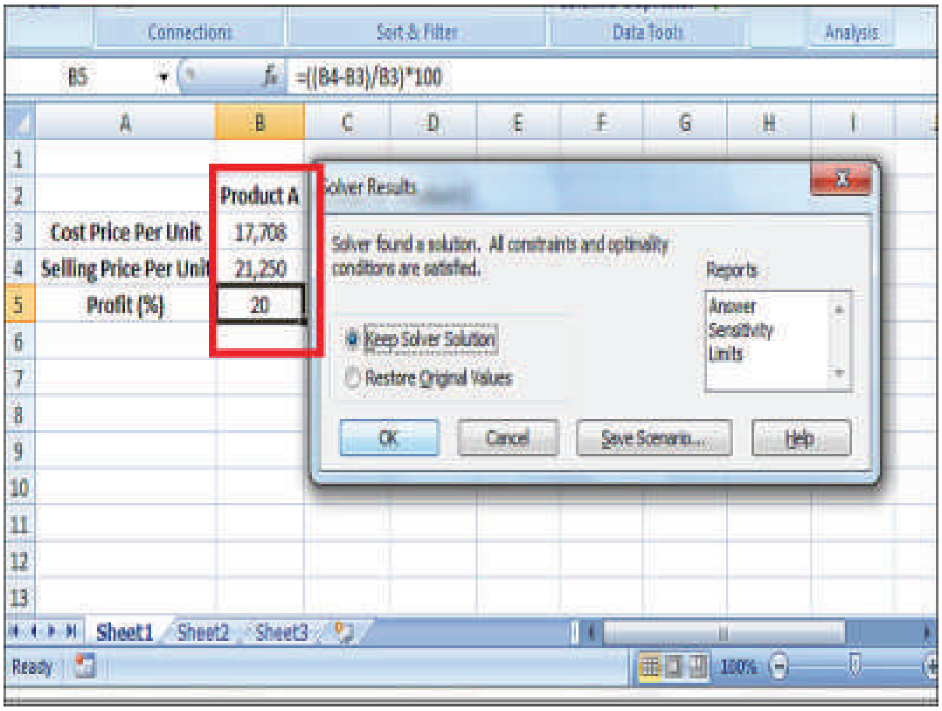
Set Target Cell: $B$5

Equal To: ○ Max ○ Min [● Value of:] 20

By Changing Cells: $B$3:$B$4

Subject to the Constraints:

STEP 3: Once you entered all the required data, click on the "Solve" button. It asks, whether you want to keep the solver solution along with some options. Select based on your requirement and click on "Ok" button.


B5 =((B4-B3)/B3)*100

	A	B
1		
2		Product A
3	Cost Price Per Unit	17,708
4	Selling Price Per Unit	21,250
5	Profit (%)	20

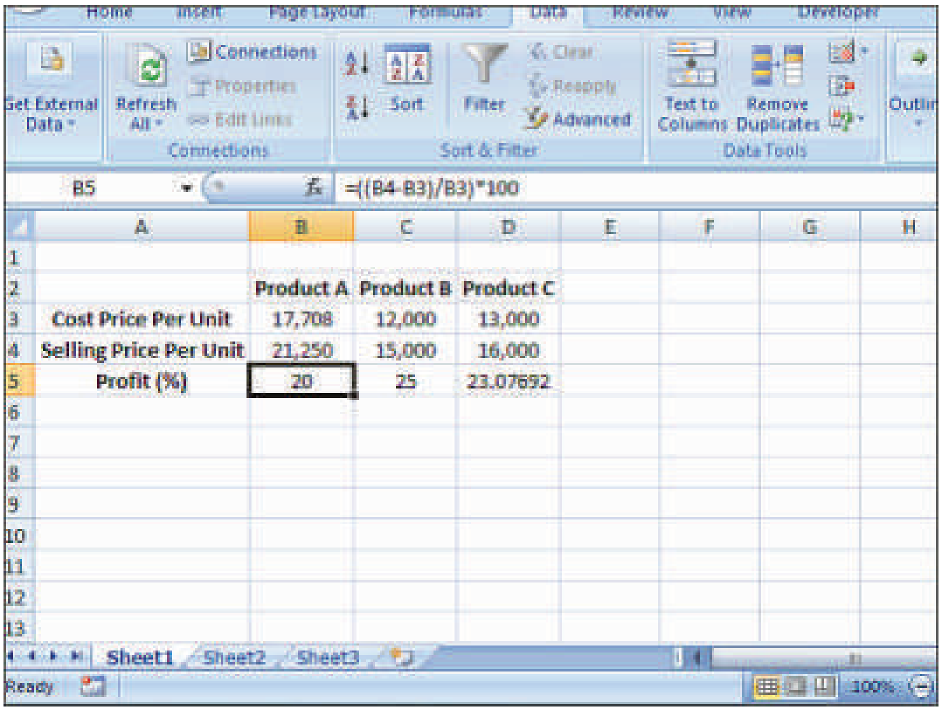
Solver Results

Solver found a solution. All constraints and optimality conditions are satisfied.

(● Keep Solver Solution) ○ Restore Original Values

Reports: Answer, Sensitivity, Limits

Now, you will see that the latest Cost Price and Selling Price has been changed to 17, 708 and 21, 250 respectively to get the 20% Profit.


B5 =((B4-B3)/B3)*100

	A	B	C	D
1				
2		Product A	Product B	Product C
3	Cost Price Per Unit	17,708	12,000	13,000
4	Selling Price Per Unit	21,250	15,000	16,000
5	Profit (%)	20	25	23.07692

This is the way to use Solver Add-in to solve equations in Excel. Explore it and you can get more out of it. Share with us how best you made use of Solver Add-in.

6.5 UNDERSTANDING OPTIMIZATION

Optimization technology is based on applied mathematics and computer science, and is widely used to help business people make better decisions. It can quickly determine how to most effectively allocate resources, automatically balancing trade-offs and business constraints. It eliminates the need to manually work out plans and schedules, so you get maximum operational efficiency. You provide the information, and IBM ILOG Optimization does the work.

WHY IBM ILOG OPTIMIZATION?

IBM is a leader in the field of Operations Research, and specifically in the discipline of optimization. We offer some of the world's most advanced optimization technologies for solving tough business and research problems—longer than anyone. Our award-winning tools and engines speak for our high standards and belief in innovation. And we're always thinking of something new.

See for yourself why more than 1,000 universities use IBM ILOG Optimization for research and teaching, and more than 1,000 commercial customers, including over 160 of the Global 500, use IBM ILOG Optimization in some of their most important planning and scheduling applications.

KEY BENEFITS

- **Maximum operational efficiency**: Improve utilization for any sort of resource: capital, personnel, equipment, vehicles, facilities. Assign the best resources to each task, at the best possible time.
- **Uncover solutions to the toughest challenges**: Explore alternatives in minutes. Cope with the most difficult trade-offs that nobody had ever considered. Extract the maximum yield from every resource. Cope with the toughest conflicts.
- **Measurable return on investment, fast**: See results within months, or even weeks. Costs drop, earnings increase and service improves. Customers are happier, and so are your employees. Some applications save thousands of dollars a year, and others save millions—but there is always a return on investment (ROI).
- **Applications in every industry**: Optimization is at work everywhere: manufacturing, transportation, logistics, financial services, utilities, energy, telecommunications, government, defense and retail.

HOW OPTIMIZATION WORKS

Optimization technology helps organizations make better plans and schedules.

A model captures your complex planning or scheduling problem. Then a mathematical engine applies the model to a scenario find the best possible solution.

When optimization models are embedded in applications, planners and operations managers can perform what-if analysis, and compare scenarios.

Equipped with intelligent alternatives, you make better decisions, dramatically improving operational efficiency.

OPTIMIZATION CAN HELP YOU:

- Cut operating costs
- Avoid capital expenses
- Shorten delivery times
- Offer flexible, precise customer service
- Provide personalized work schedules
- Manage risk
- Maximize profitability
- Understand future scenarios

There's no mystery to optimization. It's a straightforward process that achieves measurable results:

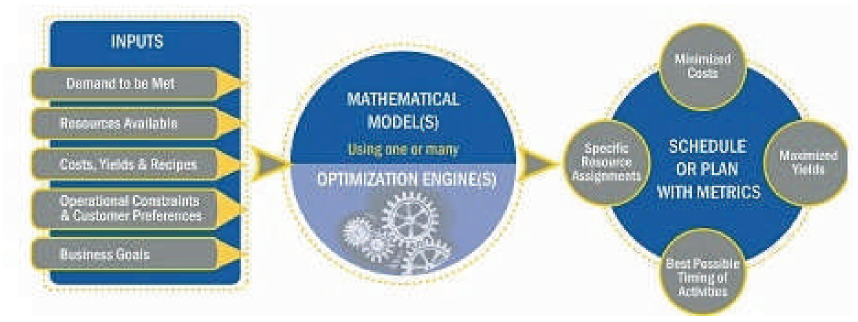
- An optimization model defines and structures your problem
- An optimization engine applies your model to data and searches for a solution
- The output is the best plan or schedule

IT ALL STARTS WITH AN OPTIMIZATION MODEL

An optimization model is a set of equations that define all of the components in a planning or scheduling problem, such as:

- Resources available
- Demand to be filled or services to be performed
- Operating and capital costs
- Yield and throughput assumptions
- Global operating constraints

- Individual operating constraints and preferences
- Goals, either individual or in weighted combination
- Key performance indicators (information about the plan)
- Decisions to be made



OPTIMIZATION ENGINES PRODUCE PLANS OR SCHEDULES

What's especially significant is the sheer amount of information that an optimization model can process. A well-built optimization model is capable of evaluating millions of possibilities, and recommending thousands of individual decisions.

An optimization model's output can take any number of forms. Examples include:

- General six month production plan
- Detailed one week production schedule
- One month workforce schedule
- Truck loading plan
- Set of routes to deliver a day's worth of goods or services
- Number of trades to bring a stock index fund back into compliance
- Marketing-offer assignments for a marketing campaign
- Best loan package at the best price
- Bids to accept in a procurement management system
- When to release airplane seats or hotel nights at a lower price.

PLANS AND SCHEDULES DELIVER ASTONISHING RESULTS

Nothing can prepare you for the effect optimization can have on operational efficiency. Plans or schedules you once labored over for days appear in just minutes—without errors. After you've

integrated optimization into your planning and scheduling, your costs will be greatly reduced.
For example:

- A car manufacturer increased productivity by 30%
- Chile's two largest forest-products companies reduced their truck fleets by 30%
- A semiconductor manufacturer cut wafer-processing cycle time in half, to just 30 days
- A major airline responded to unexpected delays with efficient crew rescheduling, saving $40 million in one year
- A package-delivery company cut costs by $87 million
- A television network increased annual advertising revenue by $50 million
- An investment firm cut transaction costs by $100 million
- A major consumer packaged goods (CPG) manufacturer dramatically increased the direct loading of trucks off its packaging lines

Often, optimization uncovers decisions you might never have considered. No one can analyze so many options, so fast. Optimization's speed gives you time to experiment with different assumptions. You're free to study a range of scenarios, applying your judgment to all your options.

What is SPSS?

SPSS is a Windows based program that can be used to perform data entry and analysis and to create tables and graphs. SPSS is capable of handling large amounts of data and can perform all of the analyses covered in the text and much more. SPSS is commonly used in the Social Sciences and in the Business World, so familiarity with this program should serve you well in future. SPSS is updated often. This document was written around an earlier version, but the differences should not cause any problems. If you want to go further and learn much more about SPSS, I strongly recommend ***Discovering Statistics using SPSS.***

OPENING SPSS

Depending upon how the computer you are working on is structured, you can open SPSS in one of the following two ways:

1. If there is SPSS shortcut like this SPSS on the desktop, simply put the cursor on it and double click the left mouse button.

2. Click the left mouse button on the start button on your screen, then put your cursor on Programs or All Programs and left click the mouse. Select SPSS 17.0 for Windows by clicking the left mouse button. Either approach will launch the program.

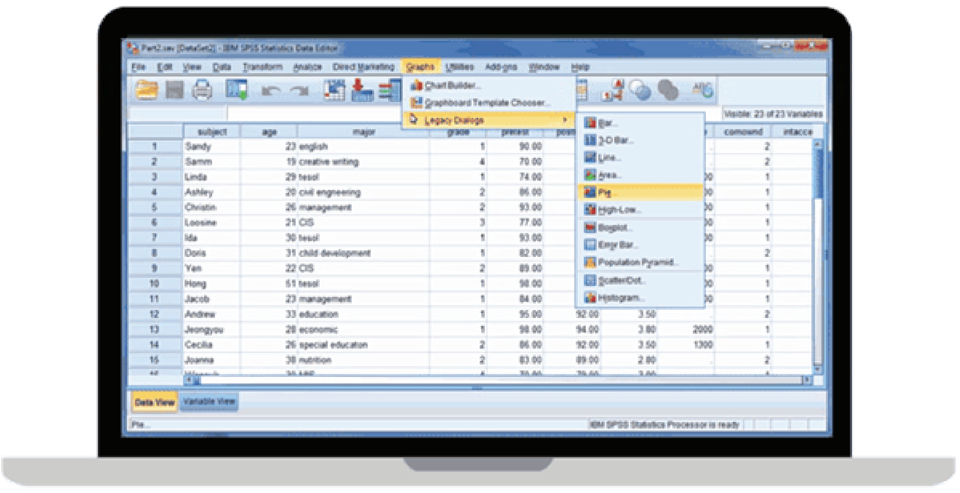
You will see a screen that looks like the image on the next page. The dialog box that appears offers choices of running the tutorial, typing in data, running queries or opening an existing data source. The Window behind this is the Data Editor Window which is used to display the data from whatever file you are using. You could select any one of the options on the start-up dialog box and click OK, or you could simply hit Cancel. If you hit Cancel, you can either enter new data in the blank Data Editor or you could open an existing file using File Menu bar as explained later.

LAYOUT OF SPSS

The Data Editor Window has two views that can be selected from the lower left hand-side of the screen. Data Views is where you see the data you are using. Variable View is where you can specify the format of your data when you are creating a file or where you can check the format of a pre-existing file. The Data in the Data Editor is saved in a file with the extension .sav.

The other most commonly used SPSS Window is the SPSS Viewer window which displays the output from any analyses that have been run and any error messages. Information from the Output Viewer is saved in a file with the extension .spo.

On the **File Menu**, click **Open** and select **Output**. Select *appendixoutput.spo* from the files that can be found at http://www.uvm.edu/~dhowell/fundamentals7/SPSSManual/SPSSL ongerManual/DataForSPSS/. Click **OK.** The following will appear. The left-hand-side is an outline of the output in the file. The right side is the actual output. To shrink or enlarge either side put your cursor on the line that divides them. When the double headed arrow appears, hold the left mouse button and move the line in either direction. Release the button and the size will be adjusted.


Part2.sav [DataSet2] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help
Chart Builder...
Graphboard Template Chooser...
Legacy Dialogs
Bar...
3-D Bar...
Line...
Area...
Pie...
High-Low...
Boxplot...
Error Bar...
Population Pyramid...
Scatter/Dot...
Histogram...
Visible: 23 of 23 Variables
Data View
Variable View
Pie...
IBM SPSS Statistics Processor is ready

	subject	age	major	grade	pretest	posttest	compound
1	Sandy	23	english	1	90.00		2
2	Samm	19	creative writing	4	70.00		2
3	Linda	29	tesol	1	74.00		1
4	Ashley	20	civil engineering	2	86.00		1
5	Christin	26	management	2	93.00		1
6	Loosine	21	CIS	3	77.00		1
7	Ida	30	tesol	1	93.00		1
8	Doris	31	child development	1	82.00		2
9	Yan	22	CIS	2	89.00		1
10	Hong	51	tesol	1	98.00		1
11	Jacob	23	management	1	84.00		1
12	Andrew	33	education	1	95.00	92.00	2
13	Jeongyou	28	economic	1	98.00	94.00	1
14	Cecilia	26	special education	2	86.00	92.00	1
15	Joanna	38	nutrition	2	83.00	89.00	2

Finally there is the Syntax window which displays the command language used to run various operations. Typically, you will simply use the dialog boxes to set up commands, and would not see the Syntax window. The Syntax window would be activated if you pasted the commands from the dialog box to it, or if you wrote your own Syntax-something we will not focus here. Syntax files end in the extension .sps.

SPSS MENUS AND ICONS

Review the options listed under each Menu Bar by clicking them one at a time. Follow the given descriptions.

File includes all of the options you typically use in other programs, such as open, save and exit. Notice, that you can open or create multiple types as illustrated on the right of the dialog box.

Edit includes the typical cut, copy and paste commands, and allows you to specify various options for displaying data and output.

Click on **Options**, you will see the dialog box to the left. You can use this to format the data, output, charts etc. These choices are rather overwhelming, and you can simply take the default options for now.

View allows you to select the toolbars you want to show, select font size, add or remove the gridlines that separate each piece of data, and to select whether or not to display your raw data or the data labels.

Data allows you to select several options to change current variables. For example, you can change continuous variables to categorical variables, change scores into rank scores, add a constant to variables etc.

Analyse includes all of the commands to carry out statistical analyses and to calculate descriptive statistics.

Graphs includes the commands to create various types of graphs including box plots, histograms, line graphs and bar charts.

Utilities allows you list file information which is a list of all variables, their labels, values, locations in the data file and type.

Add-ons are programs that can be added to the base SPSS package.

Windows can be used to select which window you want to view (for example, Data Viewer, Data Editor etc).

Help has many useful options including the link to SPSS homepage, a statistics coach and a Syntax guide. Using topics, you can use the index option to type in any key word and get a list of options, or you can view these categories and sub categories available under contents. This is an excellent tool and can be used to troubleshoot most problems.

The **Icons** directly under the Menu bar provide shortcuts to many common commands that are available in specific menus.

Place your cursor over these **Icons** for a few seconds, and the description of the underlying command will appear.

EXITING SPSS

To close SPSS either you can click on the **close** button, located on the upper right hand corner of the Screen or Select **Exit** from the **File** Menu. Choose one of these approaches.

A dialog box will appear for every open window asking you if you want to save it before exiting. You almost always want to save data files. Output files may be large, so you should ask yourself if you need to save them or if you simply want to print them. Click **No** on the dialog box since we do not have any new files or changed files to save.

LINDO (Linear, Interactive, Discrete Optimiser): Is a software package for linear programming, integer programming, non-linear programming, stochastic programming and global optimisation.

LINDO also create "What's Best!" which is an add-in for linear, integer and non-linear optimisation. First released for Lotus 1-2-3 and later also for Micro Soft Excel.

LINDO can be used:
1. To solve interactive linear, quadratic, general integer and zero-one integer programming programs up to 500 rows and 1000 columns.
2. To perform sensitivity analysis and parametric programming.

HOW TO RUN LINDO

To start an interactive session, type *lindo.*

When the program is ready for you to type a LINDO command, it displays its prompt character ":". Commands may be typed in upper or lower case. The only exception is when you type a file name since UNIX filenames are case-sensitive.

To obtain interactive help, type *help.*

To terminate LINDO, type *quit.*

LINDO displays all of its output without stopping. If you want, LINDO to pause after every 24 lines of output type, *page 24.*

ACCESSING UNIX DISK FILES

Many LINDO commands, such as take, save, retr, dvrt, and rvrt, read from or write information to UNIX disk files. The general form of these commands is
LINDO-command-name   filename

When you omit the filename, LINDO prompts you with the message

FILENAME

The filename you type must match the name of file exactly. Case is important, therefore you may not refer to a file named transport. model as TRANSPORT.MODEL. The filename cannot use name-expansion such as ~smith/LPmodels/prob1, since "~smith" will be rejected. However, if user "smith" were in his home directory, he could refer to that file as LPmodels/prob1.

TYPICAL COMMANDS

Take xmplfile - reads a file named "xmplfile" in the current directory which contains LINDO commands or a model formulation. The "take" command is typically used to input files created by an editor like vi or pico. Long lines in this file that must continue on subsequent lines can be broken at any place and immediately continued on the next line. For example, max 2a + 3.5b + 4.1c - 2.6d + 3.3e - 6.2f + 8.3 g + 4.6h - 7.7j + 5.6k
st
a + b + c + d + e + f + g + h + j + k □ 1 etc.
Lines in this file may not exceed 71 characters.

Save model 2

Saves the current model formulation in the UNIX file named *model2* in the current directory.

retr model2

Retrieves the previously saved model stored in the UNIX file *model2* in the current directory.

dvrt output3
Diverts LINDO's subsequent output from the terminal screen to the UNIX file named *output3.* This is useful for saving the results for later printing using the qpr command. For this example, you would later type the UNIX command
qpr -q smips output3
to print the output in Smith Hall.
rvrt
(revert) causes the subsequent output to be directed to the screen again.

SAMPLE INTERACTIVE SESSIONS

The following three examples illustrate how to:
1. To state and solve a linear programming problem, and save the formulation in a disk file;

2. Recover the formulation in a later session, modify the model and find the optimal solution and
3. State and solve a zero-one integer programming problem.

LINDO's responses are displayed in upper-text; the user commands and responses are in lower-case.

The average optimization software reviewed in *OR/MS Today* has two parts: the graphical user interface (GUI) and the mathematical engine. LINDO Application Programming Interface (API) is solely an engine that has been purposely separated from the GUI. This makes LINDO API a useful tool to a very focused set of Operations Research/Management Science practitioners — namely, those who write their own computer code.

(In this article, we use the word programming in its classical sense to mean the act of planning activities for a large organization using mathematical optimization. A program is a mathematical model of such a situation. Code is used to describe instructions to a computer in any of the common compiled languages: C, C++, Fortran, Java, etc.)

LINDO API, from Chicago-based LINDO Systems, Inc., is a library of functions that may be called from within many coding environments. It offers well-organized data structures and powerful linear, mixed integer and quadratic solvers in a linked library. But LINDO API was not intended for those who want to point and click their way through a model to a solution.

The intended users of LINDO API are those who wish to access from within their own application the same linear, mixed integer, and quadratic solver engines that are available from other LINDO software. Such users might have their own proprietary modeling environment, or they might be coding a large algorithm that has need for such solvers within, or they might need greater control of the solvers than they have in the other LINDO software.

LINDO API is supported on Windows, Linux and Solaris operating systems. In addition to standard hardware requirements (see Product Information box), one needs a coding environment that can link to external libraries. Most C, C++, Fortran, Java and Visual Basic environments fit this need. Alternatively, LINDO API can be called from MATLAB.

My own experience with LINDO API has been positive. For two years, LINDO Systems has allowed my research group to use LINDO API inside our own software, CPA. This software implements a generalized cutting plane algorithm for two stage stochastic linear programs. You can try out CPA at the NEOS site: www-neos.mcs.anl.gov.

SOLVERS

With LINDO API, what's under the hood is all you get. Three solvers are available: a primal simplex solver, a dual simplex solver and a barrier solver. These solvers are simply the engine from the software LINDO, which offers a graphical front end as well. The barrier solver can handle quadratic objective and constraint functions, while the others are limited to linear functions. The model input data type for each solver is the same, so changing from one solver to another is a simple matter of changing the function call. Each solver can operate within a branch and bound solver for solving mixed integer programs.

6.6 DATA STRUCTURES AND INPUT FORMATS

The basic data structures used within LINDO API are flexible, easy to interface and intuitive. The topmost structure is a modeling environment, called LSenv. The environment contains one or more model structures, called LSmodel. Each model is a single linear or quadratic program, complete with model options, data and eventually solutions. The macro-data structure is well thought out and implemented, so that many options may be set globally in the environment wrapper, or superceded at the model level.

Model data (the actual LP or QP coefficients) may be placed directly into memory by the calling code, or specified in a file in MPS or LINDO format. Direct entry involves placing your data into a sparse matrix and vector representation that is easy to use and well explained in the documentation. The internal structures appear to make efficient use of computer memory. The software can also write LP (primal or dual) data out in MPS or LINDO format.

Users of MATLAB will be glad to know that all LINDO API functions may be called from within MATLAB by using the MATLAB executable MXLINDO. Unfortunately, LINDO API cannot currently interface with mathematical modeling languages like AMPL or GAMS.

EXAMPLES OF FUNCTION CALLS

The user of LINDO API has many functions that may be called from within a coding environment. The housekeeping routines — license handling, allocation, error handling — are numerous and well organized. It is easy to use quality coding techniques when using LINDO API.

Figure 1 shows a minimal coding example, where the environment is created, the data is loaded, and the LP is solved. In addition to these basic calls, LINDO API provides a group of functions for modifying the model. One may add, delete or modify any of the model features: constraints, variables, quadratic data, integer data, right hand side coefficients, etc. Warm starts can be implemented by specification of a starting basis.

Minimal Coding Example

```
/* myexample.c

A C code example for solving the following LP:

Maximize 3 x1 + 4 x2
subj. to x1 + x2 <= 6
x1 <= 4
x1 + 2 x2 <= 11
-x1 + x2 >= -2
x1 >= 0
x2 >= 0

This is meant to be a minimal example, with very little with respect
to error checking, memory management, etc.

*/

#include <stdlib.h>
#include <stdio.h>

/* LINDO API header file */
#include "lindo.h"
```

```
/* license.h must be edited to include the license key that came with
your software */
#include "license.h"

/* main entry point */
int main()
{
int nErrorCode;

/* Number of constraints */
int nM = 4;

/* Number of variables */

int nN = 2;

/* declare an instance of the LINDO environment object */
pLSenv pEnv;

/* declare an instance of the LINDO model object */
pLSmodel pModel;

/*
*>>> Step 1 <<< Create a LINDO environment.
*/
pEnv = LScreateEnv ( &nErrorCode, MY_LICENSE_KEY);

/*
*>>> Step 2 <<< Create a model in the environment.
*/
```

Minimal Coding Example

```
/* myexample.c

A C code example for solving the following LP:

Maximize 3 x1 + 4 x2
subj. to x1 + x2 <= 6
x1 <= 4
x1 + 2 x2 <= 11
-x1 + x2 >= -2
x1 >= 0
x2 >= 0
```

```
This is meant to be a minimal example, with very little with
respect to error checking, memory management, etc.

*/

#include <stdlib.h>
#include <stdio.h>

/* LINDO API header file */
#include "lindo.h"

/* license.h must be edited to include the license key that
came with your software */
#include "license.h"

/* main entry point */
int main()
{
int nErrorCode;

/* Number of constraints */
int nM = 4;

/* Number of variables */
int nN = 2;

/* declare an instance of the LINDO environment object */
pLSenv pEnv;

/* declare an instance of the LINDO model object */
pLSmodel pModel;

/*
*>>> Step 1 <<< Create a LINDO environment.
*/
pEnv = LScreateEnv ( &nErrorCode, MY_LICENSE_KEY);

/*
*>>> Step 2 <<< Create a model in the environment.
*/
pModel = LScreateModel ( pEnv, &nErrorCode);

/*
* >>>> Step 3 <<< Specify the linear portion of the model.
*/

/* The direction of optimization */
```

```
int objsense = LS_MAX;

/* The objective's constant term */
double objconst = 0.;

/* The objective's linear term */
double c[2] = { 3., 4. };

/* The right-hand sides of the constraints */
double rhs[4] = { 6.0, 4., 11., -2. };

/* The constraint types */
char contype[4] = { 'L','L','L','G' };

/* The number of nonzeros in the constraint matrix */
int Anz = 7;

/* The indices of the first nonzero in each column */
int Abegcol[3] = { 0, 4, Anz };

/* The last entry is always the number of nonzeros */

/* The length of each column. We can set this to NULL if we
do not expect to add rows later. */

int *Alencol = NULL;

/* The nonzero coefficients by column */
double A[7] = { 1., 1., 1., -1., 1., 2., 1. };


/* The row indices of the nonzero coefficients */
int Arowndx[7] = { 0, 1, 2, 3, 0, 2, 3 };


/* By default, all variables have a lower bound of zero
* and an upper bound of infinity. Therefore pass NULL
* pointers in order to use these default values. */
double *lb = NULL, *ub = NULL;

/* Pass the linear portion of the data to problem structure
* by a call to LSloadLPData() */

nErrorCode = LSloadLPData( pModel, nM, nN, objsense,
objconst,c, rhs, contype, Anz, Abegcol, Alencol, A,
Arowndx, lb, ub);

/* >>>Step 4<<< Pass the integrality restriction to problem
```

```
structure
* by a call to LSloadMIPData() */
char vartype[14] ={ 'C','I'};

/* C means continuous, I means integer, B means binary */
nErrorCode = LSloadMIPData(pModel, vartype);

/*
* >>> Step 5 <<< Invoke the Branch & Bound solver*/
nErrorCode = LSsolveMIP( pModel, NULL);

/* >>> Step 6 <<< Retrieve the MIP solution */
int i;
double x[2], MipObj;

/* Get the value of the objective and solution */
LSgetMIPSolution( pModel, &MipObj, x) ;

printf ("*** Optimal Objective Value= %f\n", MipObj);
for (i = 0; i < nN; i++)
printf( "Optimal solution for variable %d: %5.2f \n", i+1, x[i]
);
printf ("\n");

* >>> Step 7 <<< Delete the LINDO environment */
nErrorCode = LSdeleteEnv( pEnv);
}
```

For those working on very large codes, the callback feature may be useful. If this feature is set, the solver periodically checks in with the calling function, at which time the current status may be probed, adjustments made or the solver terminated.

6.7 INSTALLATION AND TECHNICAL SUPPORT

The installation of LINDO API is straightforward on one hand and tricky on the other. On a Windows system, an install wizard makes installation easy. In Linux, a README file takes the user through an easy three-step installation process. LINDO has certainly done their part to make the setup of LINDO API easy.

However, there is a certain level of difficulty inherent in the use of callable libraries in general. The problems usually arise during the linking phase, when the compiler tries to find and use the LINDO libraries. Here, the user must wade through the swamp of compiler settings, environment variables and standard library version compatibility to try to find the magic setup that works. This can be daunting for the inexperienced code warrior.

The good news is that LINDO Systems provides excellent technical support. There is a real phone number that is answered by a real person (do I sound jaded?), and in a single transfer the caller finds that he/she is talking to a real developer of LINDO API. The service is prompt and effective. The developers even listen to suggestions. Last spring, I suggested a small change in the format of one of the function calls. It was implemented in the next few months.

DOCUMENTATION

The documentation that accompanies LINDO API is extensive and well written. Each callable function and parameter is described, and several examples of calling code are given. The examples show a variety of calling languages (C, C++, Java, Visual Basic and MATLAB). The examples given in the documentation are supposed to match files that are distributed with the library. However, some of these (probably the accompanying files) have been changed, so that the user should treat the accompanying files as additional examples, rather than exact implementations of the examples in the documentation. In addition, appendices on MPS and LINDO file formats are provided.

CONCLUSION

LINDO Systems has put forward a solid competitor in the callable optimization library market. LINDO API does not have as many features as some of its competitors (e.g. stochastic optimization solver and parallel computing), but if you only want a basic LP, MIP or QP solver, you can get a great value with LINDO API.

6.8 EXERCISES

Q1. What are the different software’s available for OR? Explain two of them.

Q2. How does solver add-in is used to solve OR problems?

Q3. What is SPSS and how does it work?

Q4. What is LINDO? and Describe the working of LINDO software.

❖❖❖❖