Integer linear programming

An integer programming problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers. Integer programming problem are a special class of linear programming where all are some of the variable in the optimal solution are restricted to non-negative integer value. Optimal solution

Maximize 200x1 + 300x2

Constraint 2x1 + 4x2+ ≤ 63

3x1 + 3x2 + ≤ 42

x1 + x2 ≥ 0 x1 x2 ,integer

**simplex tableue optimal solution**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | **x1** | | **x2** | **s1** | | **s2** | | **bi** |
| **S1** | 2 | | 4 | 1 | | 0 | | 17 |
| **S2** | 3 | | 3 | 0 | | 1 | | 15 |
| **Cj** | 200 | 300 | | | 0 | | 0 | | |
| **sloution** | 0 | 0 | | | 17 | | 15 | | |
| J | 200 | 300 | | |  | | 0 | | |

Linear programming problem are based on assumption that the decision variable are continuous with the result that they may take frictional value in the optimal solution e.g. 143/5 unit of product. However many situation in real life may not allow for this and warrant for the decision variable to take only integer value. For example in production, manufacturing is frequently scheduled in term of lots, batches or production runs. In allocation of goods, a shipment of goods involves discrete number of trucks, wagon or aircraft and fractional values are meaningless.

Types of integer programming

Pure integer programming.

An integer programming in which all variable are required to be integer is called pure integer.

Maximize 20x1 + 8x2

Constraint 5x1 + 7x2+ ≤ 63

2x1 + 3x2 + ≤ 42

x1 + x2 ≥ 0 x1 x2 ,integer

Both of these value in the optimal solution should be integer value for example x1 is 12 and x2 is 15. These values should not be like 15.5 and 12.5 in the optimal solution.

Mixed integer programming.

An integer programming in which some variable are required to be integer is called mixed integer programming.

Maximize 20x1 + 8x2

Constraint 5x1 + 7x2+ ≤ 63

2x1 + 3x2 + ≤ 42

x1 + x2 ≥ 0 x1,integer

Value of only x1 should be integer in the optimal solution. If we see there is only x1 should be integer.

Zero-one Model Integer programming.

A special type of integer programming problem is a case where the value of decision variable are limited to logical variable like Yes or No, Matched or No-Matched which are symbolized by the value 0 and 1.

Maximize 20x1 + 8x2

Constraint 5x1 + 7x2+ ≤ 63

2x1 + 3x2 + ≤ 42

4x1 + 3x2 ≤ 30

x1 + x2 ≥ 0 x1=0 (if investment is not made is made)

X2=1(if investment is made is made)

For example there are 3 project and we want to select 2 of them. When we are not invest in any project we will assign zero to that project and assign one to those in which we want to invest.

Applications of integer programming.

**Traveling sales man problem.**

This means that salesman can find shortest tour to visit all selected city. For example salesman wants to visit five selected cities. What root he must follow to minimize the distance. Through integer programming we can solve this problem.

**Capital budgeting.**

Frequently companies face the situation of selecting one or more research development projects or other investment opportunities among several competing projects. Which project should to be select and which not. This problem can be solved through integer linear programming.

**Facility location.**

Another application of integer programming is facility location. Such a problem to decide where to locate a limited number of facilities such as plant and warehouses. This can be also solve through integer programming.

**Fixed charge problem**

Many problems in real life involve a combination of fixed and variable costs. The fixed cost is incurred only if certain projects are undertaken or certain capacity level is exceeded. This problem can be solved through integer linear programming.

Now we want to solve one example of salesman travelling problem

**Salesman travelling problem and solution**

A sales man must travel city to city to maintain his account. This week he has to leave his home base and visit four other cities. The home city is city A. use the assignment method to determine the tour that will minimize the total distance of visiting all cities. The total distance of visiting of all cities and returning is given in the table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | − | 375 | 600 | 150 | 190 |
| **B** | 375 | − | 300 | 350 | 175 |
| **C** | 600 | 300 | − | 350 | 500 |
| **D** | 160 | 350 | 350 | − | 300 |
| **E** | 190 | 175 | 500 | 300 | − |

**Solution**: for this solution we will use assignment model to solve this problem. Not all the problems can be solving through this model but this problem can be solve through assignment model.

**First step:** Put M to prohibit travelling from particular city to itself. For example from Peshawar to Peshawar. It is meaningless. From A to A.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 375 | 600 | 150 | 190 |
| **B** | 375 | M | 300 | 350 | 175 |
| **C** | 600 | 300 | M | 350 | 500 |
| **D** | 160 | 350 | 350 | M | 300 |
| **E** | 190 | 175 | 500 | 300 | M |

**Step 2:**  minimum value in the row subtract from all the element in the row for example

Minimum value in the first row is 150. 375-150, 600-150, 150-150,190-150. In second row minimum value is 175.375-175,300-175,-350-175,175-175. This process is for all the rows in the table. When we complete this we will obtain another table which is follow.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 225 | 450 | 0 | 40 |
| **B** | 200 | M | 125 | 175 | 0 |
| **C** | 300 | 0 | M | 50 | 200 |
| **D** | 0 | 190 | 190 | M | 140 |
| **E** | 15 | 0 | 325 | 125 | M |

**Step 3:** in this step we will under line all those rows and columns which have zero. For this purpose if the row wise zeros are more than column wise then we will first underline row wise zeros. In the above table if we see there is only one column which has 2 zeros. So first we will under line column B. then underline those rows which have zero.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 225 | 450 | 0 | 40 |
| **B** | 200 | M | 125 | 175 | 0 |
| **C** | 300 | 0 | M | 50 | 200 |
| **D** | 0 | 190 | 190 | M | 140 |
| **E** | 15 | 0 | 325 | 125 | M |

Now if we see the draw lines are less than rows and columns in number. Columns and rows are 5 and draw lines are 4. The requirements of this model are that the draw line should be equal to rows and column. So we need for more work to equalize both draw lines and rows & column.

**Step 4:** now we select the smallest value from the uncovered values which is 15. Now we will subtract 15 from all uncovered elements. Like this 300-15,50-15,200-15,125-15,325-15,15-15. After this we will add 15 to all intersection points. 225+15,190+15 we will obtain the following table. And again under line rows and columns which have zero follow the above mentioned rules.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 240 | 325 | 0 | 40 |
| **B** | 200 | M | 0 | 175 | 0 |
| **C** | 285 | 0 | M | 35 | 185 |
| **D** | 0 | 205 | 65 | M | 140 |
| **E** | 0 | 0 | 185 | 110 | M |

Again if we see the draw lines are less than rows and columns. So we select smallest value from those elements which are not underlined. This value is 35. So we subtract this value from all those element which are not underlined. And add to those elements which are located on intersection points. Intersection points and not underlined element are explained in the previous page. So we will obtain the following table. And again underline the rows and column which have zeros.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 275 | 325 | 0 | 40 |
| **B** | 235 | M | 0 | 175 | 0 |
| **C** | 285 | 0 | M | 0 | 150 |
| **D** | 0 | 205 | 30 | M | 105 |
| **E** | 0 | 0 | 150 | 75 | M |

Again the drawn lines are less than rows and column. We again apply the previous procedure to get drawn lines equal to rows and columns. We will select smallest value which is 30 and subtract from all uncovered elements and add to the intersection points. And again underline the rows and column which have zeros. We will obtain the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 275 | 295 | 0 | 10 |
| **B** | 265 | M | 0 | 205 | 0 |
| **C** | 285 | 0 | M | 0 | 120 |
| **D** | 0 | 205 | 0 | M | 75 |
| **E** | 0 | 0 | 150 | 75 | M |

**Step 5:** we have obtained drawn lines equal to rows and column. Now we will select zero from each rows and columns. We will start from the first row which is not underlined. When we selected zero in first row all other zero in the same row and column should be crossed. When we crossed zero in column D we automatically select zero in row C and cross zero in in B column. Procedure for selecting zeros.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | M | 275 | 295  0 | 0 | 10 |
| **B** | 265 | M | 0 | 205  0 | 0 |
| **C** | 285 | 0  O | M | 0 | 120 |
| **D** | 0 | 205 | 0  O | M | 105 |
| **E**  O | 0 | 0 | 120 | 75 | M |

**Step 6:** now we go back to original first table which have the distance values of different cities. In the last table we have selected zeros. Now we select those values which exist on the place of zeros which we have selected. Which are underlined values in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **From city** | **A** | **B** | **C** | **D** | **E** |
| **A** | − | 375 | 600 | 150 | 190 |
| **B** | 375 | − | 300 | 350 | 175 |
| **C** | 600 | 300 | − | 350 | 500 |
| **D** | 160 | 350 | 350 | − | 300 |
| **E** | 190 | 175 | 500 | 300 | − |

**Step 7:** The root which minimize the distance is A-D,B-E,C-B,D-C,E-A. When we combine these we get the following root. A-D-C-B-E-A

This step can be check in the **step 5** table.

Now put the value A=150 B=175 C=300 D=350 E=190

150+350+300+175+190=1165KM

The salesman must travel to the above selected cities to minimize the distance of his visit to all cities.

**Second example**

The owner of readymade garments store sell two type of premium shirt known as zee shirt and star shirts. He makes a profit of Rs 200 and Rs 300 per shirt of zee and star shirt. He has two tailor A and B. tailor A can devote a total of 17 hours per day. While tailor B can devote 15 hours a day. Both type of shirts are stitched by both the tailor. the time needed for stitching zee shirt is 2 hour by tailor A and 3 hour by tailor B. a star shirt require 4 hour of tailor A and 3 hour by tailor B. how many shirts of each should be stitched to maximize daily profit. Solve it as integer problem.

**Solution:** this problem will be solving through simplex method.

**Step 1:** define objective function and constraint function which is used in linear programming

maximize 2oox1 + 300x

subject to

2x1 + 4x2+ ≤ 17

3x1 + 3x2 + ≤ 15

x1 + x2 ≥ 0, x1 integer

In this problem we want the value of x1 to be integer value.

**Step 2:** now we add slack variable to the objective and constraint function.

Z =2oox1 + 300x+0S1+0S2

subject to

2x1 + 4x2+ S1+0S2= 17

3x1 + 3x2 + 0S1+S2= 15

There are two variable x1 and x2. So we will add to slack variable s1 and s2. S1 show unused time by tailor A. s2 show unused time by tailor B.

**Step 3:** now we put the above constraint function in the table which is called standardization.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | **x1** | | **x2** | | **s1** | | | **s2** | | **bi** |
| **S1** 0 | 2 | | 4 | | 1 | | | 0 | | 17 |
| **S2** 0 | 3 | | 3 | | 0 | | | 1 | | 15 |
| **Cj** | | 200 | | 300 | | 0 | 0 | |
| **sloutio** | | 0 | | 0 | | 17 | 15 | |
| J | | 200 | | 300 | | 0 | 0 | |

(Basic variable) (Values of basic and non basic variable) (Contribution to the profit per unit)

(Right hand side of the constraint)

This table will be used to solve the above problem. This problem can be solved through simplex method.

**Step 4:** now we will find key row and key column to solve the problem because this is the essential step. To find key column we will select highest value in the J row which is 300.when we find the key column it made easy for us to find key row.

To find key row see next page.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | | **x1** | **x2** | | | **s1** | **s2** | | **bi** |
| **S1** 0 | | 2 | 4\* | | | 1 | 0 | | 17 4.25 |
| **S2** 0 | | 3 | 3 | | | 0 | 1 | | 15 5 |
| **Cj** | 200 | | | 300 | 0 | | 0 |
| **sloutio** | 0 | | | 0 | 17 | | 15 |
| J | 200 | | | 300 | 0 | | 0 |

Key column

To find key row divide **bi** by key column element. 17/4 and 15/5 we will get this value. Now the key row is that which have smallest quotient. The key column is **incoming variable** and key row is **outgoing variable**.

**Step 5:** now we will divide row one elements by element in key column 4 which is symbolized by \*. We divided by 4 because this is the intersection point of key column and key row.

2/4= .5, 4/4=1, 1/4=.25, 0/4=0, 17/4=4.25.

For row 2 we will use the following procedure.

Old row 2 element – row element in key column \*correspondent replacement values= new

3 - 3 \* .5 =1.5

3 - 3 \* 1 =0

0 - 3 \* 0.25 = -0.75

1 - 3 \* 0 = -1

15 - 3 \* 4.25 = 2.25

Now we will put these elements in another table which is called revised table.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | | **x1** | | **x2** | **s1** | | | **s2** | | **bi** |
| **X2 300** | | .5 | | 1 | .25 | | | 0 | | 4.25 |
| **S2**  0 | | 1.5\* | | 0 | -.75 | | | 1 | | 2.25 |
| **Cj** | 200 | | 300 | | | 0 | 0 | |
| **sloutio** | 0 | | 4.25 | | | 0 | 2.25 | |
| J | 50 | | 0 | | | -75 | 0 | |

300\*.25= 75

0-75= -75

300\*1=300

Now 300-300=0

300\*.5= 150

Now 200-150=50

In the above table s1 was out going variable and 300 x2 was in coming variable so x1 is replaced by x2 300.

The above table sows non optimal solution because only one variable is considered in solution. Which are 4.25. if we put this in the objective function we will get non optimal solution. now we put it in objective function.

Z=200X1+300X2+0S1+0S2 now put in z= 200(0)+300(4.25)+0(0)+0(2.25)= 1275

**Step 6:** to get optimal solution we will again revise the above table. For this purpose we again to find key column and key row. For key column select small element in the **j** row symbolized by which is 50. For key row we select smallest value symbolized by which is 2.25.

In the above table row 2 will be divided by key element which is symbolized by \* which is 1.5. Because this value existed on the intersection point of key column and key row. So we divide row 2 by 1.5. 1.5/1.5=1, 0/1.5=0, -0.75/1.5=-0.5, 1/1.5=.666, 2.25/1.5=1.5

For row one we will use the above procedure.

Old row one element – row element in key column \*correspondent replacement values= new

.5 - .5 \* 1 =0

1 - .5 \* 0 =1

0.25 - .5 \* -0.5 = .5

0 - .5 \* .666 = -.333

4.25 - .5 \* 1.5 = 3.5

Now we will put all these value in the revised table.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | **x1** | | **x2** | | **s1** | | **s2** | | **bi** | |
| **X2 300** | 0 | | 1 | | .5 | | -.333 | | 3.5 | |
| **X1 200** | 1 | | 0 | | -.5 | | .666 | | 1.5  200\*1=200  Now 200-200=0 | |
| **Cj** | | 200 | | 300 | | 0 | | 0  300\*1= 150  Now 300-300=0 | |
| **sloutio** | | 1.5 | | 3.5 | | 0 | | 0  300\*.5= 150  200\*-0.5= -100  150-100= 50 but we will show this in negative | |
| J | | O | | 0 | | -50 | | -33.33 | |

Same procedure

Now we have obtained optimal solution. The requirement of this question was that x1 should be integer value in the solution. But if we see in the solution the value is not integer it is fractional.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cj** | 200 | 300 | 0 | 0 |
| **sloutio** | 1.5 | 3.5 | 0 | 0 |
| J | O | 0 | -50 | -33.33 |

**Step 7:** To bring the value of x1 which 1.5 we will apply cutting plane algorithm (gormy cut) which is used to bring fractional value into integer value in the integer linear programming. For **this purpose we select first row of the table in step 6 which is**

If we divide 1/3 we will get .333 so in the table .333 is negative. To finish this negative sign we add -1 to 2 which is equal to 1.

0x1 + x2+.5S1+(-.333)S2= 3.5

0x1 + x2+.5S1+(-1+2/3)S2= 3.5

Now we collect all the integer value and transfer them to right hand side and Crosse them. Because this the requirement of gormy cut.

.5S1+2/3S2= .5 (3 +0x1 + 1x2 - 1S2)

Now we only consider the fractional part of the equation and change of the equation and add another slack variable..

-5S1-.666S2+S3= -.5

Now we add this equation to **table in step 6.**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | | **x1** | | **x2** | | **s1** | | **S2 S3** | | **bi** | |
| **X2 300** | | 0 | | 1 | | .5 | | -.333 0 | | 3.5 | |
| **S2 200**  **S3** | | 1  0 | | 0  0 | | -.5  -.5 | | .666 0  -.666\* 1 | | 1.5  -.5 | |
| **Cj** | | 200 | | 300 | | 0 | | 0 0 | |
| **sloutio** | | 1.5 | | 3.5 | | 0 | | 0 -.5 | |
| J | | O | | 0 | | -50 | | -33.33 0 | |

We will obtain the above table after adding the 3rd equation. To find key column and key row we again follow the previous procedure. For key column select minimum value in **j** row which is -33.33 . And for key row -.5 in **bi** column. Key element is -.666 symbolized by \*. Now we again divide row three by key element by -.666. For row one and row two we follow the previous procedure in same in step 5 and step 6. We will obtain revised table for solution. Which exist in the next page.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Basic** | **x1** | **x2** | | **s1** | | **S2 S3** | | **bi** | |
| **X2 300** | 0 | 1 | | .75 | | 0 -.5 | | 3.5 | |
| **X1 200**  **S2** | 1  0 | 0  0 | | -1  .75 | | 0 1  1 -1.5 | | 1.5  -.5 | |
| **Cj** | 200 | | 300 | | 0 | | 0 0 | |
| **sloutio** | 1 | | 3.75 | | 0 | | .75 0 | |
| J | 0 | | 0 | | -25 | | 0 -50 | |

Same procedure

As in step 5 and 6

Now we have obtained integer value which was the requirement of problem

If we want to get integer value of x2. We will again use cutting plane algorithm. Same procedure as previous and add 4ht slack variable.

X1= 1 x2= 3.75 now put in the objective function.

**z**= 200(1)+300(3.755)+0(0)+0(0.75)=1325