Introduction to Linear Programming (LP) and Linear Programming Problems (LPP)

Linear Programming (LP)

Linear Programming is a mathematical technique used for optimization, where the goal is to achieve the best outcome (such as maximum profit or minimum cost) in a mathematical model whose requirements are represented by linear relationships. LP is widely used in various fields such as business, economics, engineering, and military applications.

Components of Linear Programming

1. **Decision Variables: Variables that decide the output. These are the variables we want to solve for.**
2. **Objective Function: A linear function that needs to be maximized or minimized.**
3. **Constraints: A set of linear inequalities or equations representing the restrictions or limitations on the decision variables.**
4. **Non-negativity Restriction: The decision variables should be non-negative, meaning they should be greater than or equal to zero.**

Linear Programming Problem (LPP)

A Linear Programming Problem is a specific type of LP problem where the objective function and constraints are all linear.

Case Study - 1 (Maximization Problem)

KK Electronics Company manufactures two types of pen drives, Pendrive A and Pendrive B, which are used in various electronic devices. The company aims to maximize its profit from the production and sale of these pen drives. The unit selling price of Pendrive A is Rs 1500, and for Pendrive B, it is Rs 2500. Profits are directly proportional to the selling prices.

To produce these pen drives, the company needs to allocate resources such as skilled labor, unskilled labor, and raw materials. Each pendrive requires a different amount of these resources, and the company has limited availability of these resources. Additionally, there is a minimum production requirement for both types of pen drives to meet the market demand.

Problem Formulation

Given Information:

1. **Selling Price:**
   - **Pendrive A: Rs 1500 per unit**
   - **Pendrive B: Rs 2500 per unit**
2. **Resources Required:**
   - **For Pendrive A:**
     - **Skilled labor: 3 hours per unit**
     - **Unskilled labor: 2 hours per unit**
     - **Raw material: 1 unit per unit**
   - **For Pendrive B:**
     - **Skilled labor: 4 hours per unit**
     - **Unskilled labor: 3 hours per unit**
     - **Raw material: 2 units per unit**
3. **Resource Availability:**
   - **Skilled labor: 200 hours**
   - **Unskilled labor: 170 hours**
   - **Raw material: 30 units**
4. **Minimum Production Requirement:**
   - **At least 3 units of Pendrive A**
   - **At least 5 units of Pendrive B**

Formulation of the Linear Programming Problem

1. **Decision Variables:**
   - **Let x be the number of units of Pendrive A produced.**
   - **Let y be the number of units of Pendrive B produced.**
2. **Objective Function:**
   - **Maximize Profit PPP: $P=1500x+2500y$**
3. **Constraints:**
   - **Skilled Labor Constraint: $3x+4y\leq200$**
   - **Unskilled Labor Constraint: $2x+3y\leq170$**
   - **Raw Material Constraint: $x+2y\leq30$**
   - **Minimum Production Constraints: $x\geq3$ $y\geq5$**
   - **Non-negativity Constraint: $x\geq0, y\geq0$**

Explanation of Factors

1. **Decision Variables: x and y represent the number of units of Pendrive A and Pendrive B produced, respectively. These are the values we need to determine to maximize profit.**
2. **Objective Function: The goal is to maximize the profit, which is calculated based on the number of units produced and their respective selling prices. For Pendrive A, the profit contribution is Rs 1500 per unit, and for Pendrive B, it is Rs 2500 per unit.**
3. **Constraints:**
   - **Skilled Labor Constraint: The total hours of skilled labor used for producing both pen drives should not exceed the available 200 hours.**

- Unskilled Labor Constraint: The total hours of unskilled labor used for producing both pen drives should not exceed the available 170 hours.
- Raw Material Constraint: The total units of raw materials used for producing both pen drives should not exceed the available 30 units.
- Minimum Production Constraints: There is a minimum production requirement to ensure market demand is met: at least 3 units of Pendrive A and 5 units of Pendrive B must be produced.
- Non-negativity Constraint: The number of units produced cannot be negative; hence, both x and y must be greater than or equal to zero.

Implementation :

```
import pulp

# Create a Linear Programming problem
lp_problem = pulp.LpProblem("Maximize_Profit", pulp.LpMaximize)

# Decision variables
x = pulp.LpVariable('x', lowBound=0, cat='Continuous')  # Units of Pendrive A
y = pulp.LpVariable('y', lowBound=0, cat='Continuous')  # Units of Pendrive B

# Objective function
lp_problem += 1500 * x + 2500 * y, "Total_Profit"

# Constraints
lp_problem += 3 * x + 4 * y <= 200, "Skilled_Labor"
lp_problem += 2 * x + 3 * y <= 170, "Unskilled_Labor"
lp_problem += x + 2 * y <= 30, "Raw_Material"
lp_problem += x >= 3, "Minimum_Pendrive_A"
lp_problem += y >= 5, "Minimum_Pendrive_B"

# Solve the problem
lp_problem.solve()

# Print the results
print(f"Status: {pulp.LpStatus[lp_problem.status]}")
print(f"Optimal number of Pendrive A to produce: {x.varValue}")
print(f"Optimal number of Pendrive B to produce: {y.varValue}")
print(f"Maximum Profit: Rs {pulp.value(lp_problem.objective)}")
```

```
Status: Optimal
Optimal number of Pendrive A to produce: 20.0
Optimal number of Pendrive B to produce: 5.0
Maximum Profit: Rs 42500.0
```

Explanation of the Code:

1. Create a Linear Programming Problem:
   - lp_problem = pulp.LpProblem("Maximize_Profit", pulp.LpMaximize) initializes the problem as a maximization problem.
2. Decision Variables:
   - x and y are created as continuous variables with a lower bound of 0. These represent the number of units of Pendrive A and Pendrive B produced, respectively.
3. Objective Function:
   - The objective function is defined to maximize the total profit: 1500 * x + 2500 * y.
4. Constraints:
   - The constraints for skilled labor, unskilled labor, raw material, and minimum production requirements are added to the problem.
   - 3 * x + 4 * y <= 200 ensures that the total hours of skilled labor used for producing both pen drives do not exceed the available 200 hours.

- 2 * x + 3 * y <= 170 ensures that the total hours of unskilled labor used for producing both pen drives do not exceed the available 170 hours.
- x + 2 * y <= 30 ensures that the total units of raw materials used for producing both pen drives do not exceed the available 30 units.
- x >= 3 ensures that at least 3 units of Pendrive A are produced.
- y >= 5 ensures that at least 5 units of Pendrive B are produced.

5. Solve the Problem:
   - lp_problem.solve() solves the problem using the default solver.
6. Print the Results:
   - The status of the solution, optimal values for x and y, and the maximum profit are printed.

Case Study - 2 (Minimization Problem)

In the modern world, with rising food prices and the need for balanced nutrition, developing a cost-effective diet plan is crucial. Diet optimization is a practical application of Linear Programming (LP), where the goal is to determine the most economical way to meet daily nutritional requirements using available food items.

Imagine you are tasked with developing a diet plan that meets daily nutritional requirements at a minimum cost. You have two food items to choose from, each with different costs and nutritional content. The objective is to create a diet plan that fulfills the daily nutritional needs while minimizing the total cost.

Problem Statement

Imagine you are tasked with developing a diet plan that meets daily nutritional requirements at a minimum cost. You have two food items to choose from:

1. Food A:
   - Cost: $3 per unit
   - Provides 2 units of protein and 4 units of vitamins per unit.
2. Food B:
   - Cost: $1 per unit
   - Provides 1 unit of protein and 2 units of vitamins per unit.

The daily nutritional requirements are:

- At least 8 units of protein.
- At least 16 units of vitamins.

Formulation of the Linear Programming Problem

1. Decision Variables:
   - Let x be the number of units of Food A.
   - Let y be the number of units of Food B.
2. Objective Function:
   - Minimize Cost C: C=3x+1y

3. **Constraints:**
   - **Protein Constraint: $2x+1y \geq 8$**
   - **Vitamins Constraint: $4x+2y \geq 16$**
   - **Non-negativity Constraint: $x \geq 0, y \geq 0$**

```python
import pulp

# Create a Linear Programming problem
diet_problem = pulp.LpProblem("Minimize_Diet_Cost", pulp.LpMinimize)

# Decision variables
x = pulp.LpVariable('x', lowBound=0, cat='Continuous')  # Units of Food A
y = pulp.LpVariable('y', lowBound=0, cat='Continuous')  # Units of Food B

# Objective function
diet_problem += 3 * x + 1 * y, "Total_Cost"

# Constraints
diet_problem += 2 * x + 1 * y >= 8, "Protein_Requirement"
diet_problem += 4 * x + 2 * y >= 16, "Vitamins_Requirement"

# Solve the problem
diet_problem.solve()

# Print the results
print(f"Status: {pulp.LpStatus[diet_problem.status]}")
print(f"Units of Food A: {x.varValue}")
print(f"Units of Food B: {y.varValue}")
print(f"Total Cost: ${pulp.value(diet_problem.objective)}")
```

```
Status: Optimal
Units of Food A: 0.0
Units of Food B: 8.0
Total Cost: $8.0
```

Explanation

1. **Import PuLP Library:**
   - **import pulp**
2. **Create LP Problem:**
   - **diet_problem = pulp.LpProblem("Minimize_Diet_Cost", pulp.LpMinimize) creates an LP problem to minimize the cost.**
3. **Decision Variables:**
   - **x represents the number of units of Food A.**
   - **y represents the number of units of Food B.**
4. **Objective Function:**
   - **diet_problem += 3 * x + 1 * y, "Total_Cost" defines the objective function to minimize the total cost.**
5. **Constraints:**
   - **diet_problem += 2 * x + 1 * y >= 8, "Protein_Requirement" ensures the protein requirement is met.**
   - **diet_problem += 4 * x + 2 * y >= 16, "Vitamins_Requirement" ensures the vitamins requirement is met.**
6. **Solve the Problem:**
   - **diet_problem.solve() solves the LP problem.**
7. **Print the Results:**
   - **The status, number of units of each food, and total cost are printed.**

Case Study - 3 (Transportation Problem)

A dairy firm operates two milk plants with a daily production capacity of 6 million liters and 9 million liters, respectively. The firm's objective is to distribute the produced milk to three distribution centers with daily requirements of 7 million liters, 5 million liters, and 3 million liters, respectively. The firm aims to minimize the transportation cost of delivering milk from the plants to the distribution centers.

Problem Formulation

To solve this problem, we need to formulate a Linear Programming (LP) model. The decision variables, objective function, and constraints are described below:

Decision Variables:

- x11: Amount of milk transported from Plant 1 to Distribution Centre 1

- x12: Amount of milk transported from Plant 1 to Distribution Centre 2

- x13: Amount of milk transported from Plant 1 to Distribution Centre 3

- x21: Amount of milk transported from Plant 2 to Distribution Centre 1

- x22: Amount of milk transported from Plant 2 to Distribution Centre 2

- x23: Amount of milk transported from Plant 2 to Distribution Centre 3

Objective Function:

Minimize the total transportation cost:

$$Z = 2x11 + 3x12 + 11x13 + 1x21 + 9x22 + 6x23$$

(Note: Costs are in hundreds of rupees)

Constraints:

1. Supply constraints for the plants:

$$x11 + x12 + x13 \leq 6$$

$$x21 + x22 + x23 \leq 9$$

2. Demand constraints for the distribution centers:

$$x11 + x21 \geq 7$$

$$x12 + x22 \geq 5$$

x13 + x23 >=3

3. Non-negativity constraints:

xij >=0 for all i = {1,2} and j = {1,2,3}

Python Code to Solve the LP Model

```
# Create a LP minimization problem
prob = pulp.LpProblem("Milk_Transportation_Problem", pulp.LpMinimize)

# Define decision variables
x11 = pulp.LpVariable('x11', lowBound=0, cat='Continuous')
x12 = pulp.LpVariable('x12', lowBound=0, cat='Continuous')
x13 = pulp.LpVariable('x13', lowBound=0, cat='Continuous')
x21 = pulp.LpVariable('x21', lowBound=0, cat='Continuous')
x22 = pulp.LpVariable('x22', lowBound=0, cat='Continuous')
x23 = pulp.LpVariable('x23', lowBound=0, cat='Continuous')

# Define the objective function
prob += 2*x11 + 3*x12 + 11*x13 + 1*x21 + 9*x22 + 6*x23, "Total Transportation Cost"

# Define constraints
# Supply constraints
prob += x11 + x12 + x13 <= 6, "Supply_Plant_1"
prob += x21 + x22 + x23 <= 9, "Supply_Plant_2"

# Demand constraints
prob += x11 + x21 >= 7, "Demand_Distribution_Centre_1"
prob += x12 + x22 >= 5, "Demand_Distribution_Centre_2"
prob += x13 + x23 >= 3, "Demand_Distribution_Centre_3"

# Solve the problem
prob.solve()

# Print the results
print(f"Status: {pulp.LpStatus[prob.status]}")
for v in prob.variables():
    print(f"{v.name} = {v.varValue}")

print(f"Total Transportation Cost = {pulp.value(prob.objective)} hundreds of rupees")
```

```
Status: Optimal
x11 = 1.0
x12 = 5.0
x13 = 0.0
x21 = 6.0
x22 = 0.0
x23 = 3.0
Total Transportation Cost = 41.0 hundreds of rupees
```

Explanation of the Code

- Problem Definition: An instance of an LP problem is created for minimization.

- Decision Variables: Six decision variables (x11, x12, x13, x21, x22, x23) are defined with a lower bound of 0.

-Objective Function: The transportation costs are minimized.

-Constraints: The supply constraints for both plants and the demand constraints for all distribution centers are defined.

-Solution: The problem is solved using the default solver, and the optimal values for the

decision variables and the minimum transportation cost are printed.

Case study - 4 (Scheduling Problem)

A ship has three cargo holds (Forward, Centre, Aft) with specific weight and volume capacities. The goal is to load these holds with different commodities (A, B, C) in such a way that the total profit is maximized. The weight distribution must be proportional to the capacity limits to preserve the ship's trim.

Problem Formulation

To solve this problem, we need to formulate a Linear Programming (LP) model. The decision variables, objective function, and constraints are described below:

Decision Variables:

- xii: Amount of commodity \(i \) loaded into hold \(j \)

- x11: Amount of commodity A loaded into Forward hold

- x12: Amount of commodity A loaded into Centre hold

- x13: Amount of commodity A loaded into Aft hold

- x21: Amount of commodity B loaded into Forward hold

- x22: Amount of commodity B loaded into Centre hold

- x23: Amount of commodity B loaded into Aft hold

- x31: Amount of commodity C loaded into Forward hold

- x32: Amount of commodity C loaded into Centre hold

-x33: Amount of commodity C loaded into Aft hold

Objective Function:

Maximize the total profit:

Z = 60x11 + 60x12 + 60x13 + 80x21 + 80x22 + 80x23 + 50x31 + 50x32 + 50x33

Constraints:

1. Capacity constraints for each hold (in tonnes):

- Forward: x11 + x21 + x31 <=2000

- Centre: x21 + x22 + x32 <= 3000

- Aft: x31 + x23 + x33 <= 1500

2. Volume constraints for each hold (in \(m^3 \)):

- Forward: 60x11 + 50x21 + 25x31 <= 100000

- Centre: 60x12 + 50x22 + 25x32 <=135000

- Aft: 60x13 + 50x23 + 25x33 <=30000

3. Total availability constraints for each commodity:

- Commodity A: x11 + x12 + x13 <=6000

- Commodity B: x21 + x22 + x23 <= 4000

- Commodity C: x31 + x32 + x33 <= 2000

4. Weight proportionality constraints (assuming proportionality is required across the holds):

- (x11 + x21 + x31)/2000 = (x12 + x22 + x32)/3000 = (x13 + x23 + x33)/1500

5. Non-negativity constraints:

- xij >= 0 for all i = {1, 2, 3\}, j = {1, 2, 3}

Python Code to Solve the LP Model

```python
# Create a LP maximization problem
prob = pulp.LpProblem("Cargo_Loading_Problem", pulp.LpMaximize)

# Define decision variables
x11 = pulp.LpVariable('x11', lowBound=0, cat='Continuous')
x12 = pulp.LpVariable('x12', lowBound=0, cat='Continuous')
x13 = pulp.LpVariable('x13', lowBound=0, cat='Continuous')
x21 = pulp.LpVariable('x21', lowBound=0, cat='Continuous')
x22 = pulp.LpVariable('x22', lowBound=0, cat='Continuous')
x23 = pulp.LpVariable('x23', lowBound=0, cat='Continuous')
x31 = pulp.LpVariable('x31', lowBound=0, cat='Continuous')
x32 = pulp.LpVariable('x32', lowBound=0, cat='Continuous')
x33 = pulp.LpVariable('x33', lowBound=0, cat='Continuous')

# Define the objective function
prob += 60*x11 + 60*x12 + 60*x13 + 80*x21 + 80*x22 + 80*x23 + 50*x31 + 50*x32 + 50*x33, "Total Profit"

# Define constraints
# Capacity constraints in tonnes
prob += x11 + x21 + x31 <= 2000, "Tonnes_Forward"
prob += x12 + x22 + x32 <= 3000, "Tonnes_Centre"
prob += x13 + x23 + x33 <= 1500, "Tonnes_Aft"

# Volume constraints in m^3
prob += 60*x11 + 50*x21 + 25*x31 <= 100000, "Volume_Forward"
prob += 60*x12 + 50*x22 + 25*x32 <= 135000, "Volume_Centre"
prob += 60*x13 + 50*x23 + 25*x33 <= 30000, "Volume_Aft"

# Total availability constraints
prob += x11 + x12 + x13 <= 6000, "Total_A"
prob += x21 + x22 + x23 <= 4000, "Total_B"
prob += x31 + x32 + x33 <= 2000, "Total_C"

# Weight proportionality constraints
prob += (x11 + x21 + x31)/2000 == (x12 + x22 + x32)/3000, "Proportional_1"
prob += (x12 + x22 + x32)/3000 == (x13 + x23 + x33)/1500, "Proportional_2"

# Solve the problem
prob.solve()

# Print the results
print(f"Status: {pulp.LpStatus[prob.status]}")
for v in prob.variables():
    print(f"{v.name} = {v.varValue}")

print(f"Total Profit = {pulp.value(prob.objective)} Rs.")
```

```
Status: Optimal
x11 = 4.3974069e-10
x12 = 0.0
x13 = 0.0
x21 = 1600.0
x22 = 2400.0
x23 = 0.0
x31 = 0.0
x32 = 0.0
x33 = 1200.0
Total Profit = 380000.00000002637 Rs.
```

Explanation of the Code

- Problem Definition: An instance of an LP problem is created for maximization.

- Decision Variables: Nine decision variables (x11 to x33) are defined with a lower bound of 0.

- Objective Function: The total profit is maximized.

- Constraints: Constraints include capacity in tones , volume in cubic meters, total availability of each commodity, and weight proportionality.

- Solution: The problem is solved using the default solver, and the optimal values for the decision variables and the maximum profit are printed.