# MOUNTAINS OF THE MOON UNIVERSITY FACULTY OF SCIENCE TECHNOLOGY AND INNOVATION DEPARTMENT OF COMPUTER SCIENCE COURSE WORK LP GRAPHICAL

SANTO RAYERN
REG NO.2023/U/MMU/BCS/01673
LINEAR PROGRAMMING

13 February 2024

GRAPHICAL METHOD

## 1 BASIC RESOURCE ALLOCATION

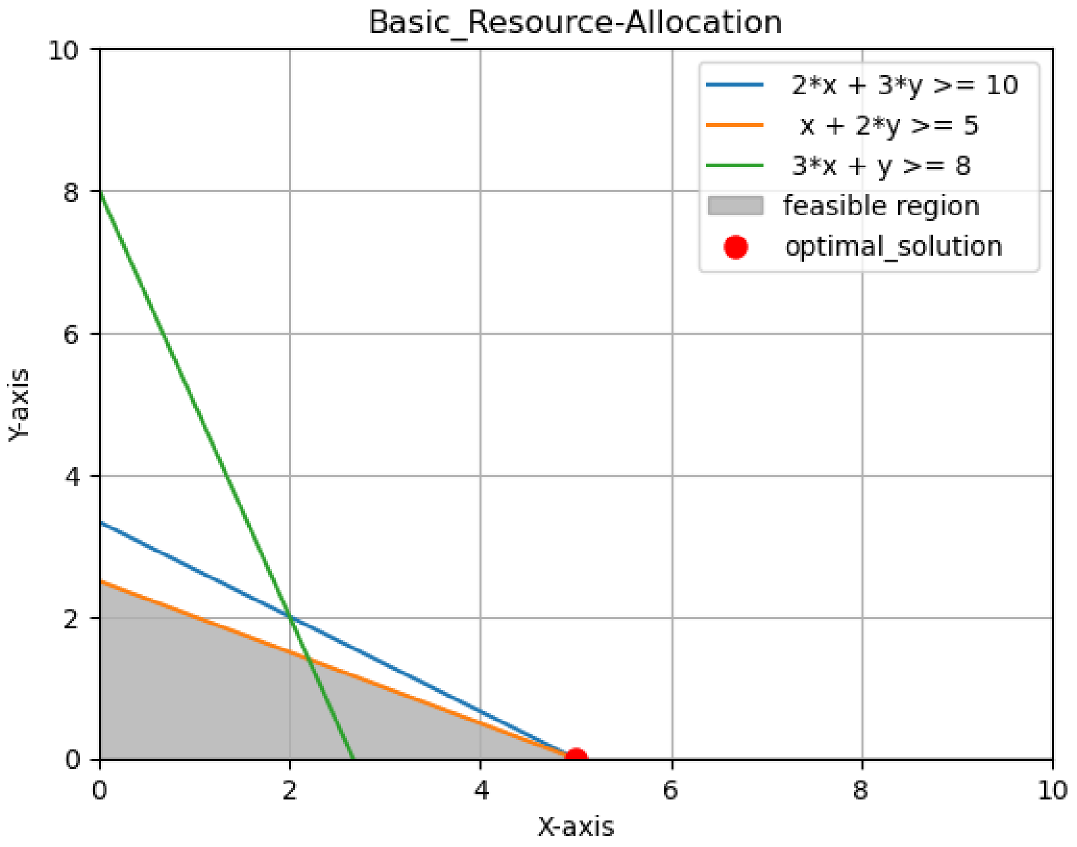
```
# importing necessary libraries:
from pulp import LpProblem,LpMinimize,LpVariable

# create a linear programming problem
model = LpProblem(name="Basic_Resource_Allocation_Optimization", sense=LpMinimize)

# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
model  += 2 * x + 5 * y, "Objective"

# Define constraints
model += 2 * x + 3 * y >= 10, "CPU"
model +=  x + 2 * y >= 5, "MEMORY"
model += 3 * x +  y >= 8, "STORAGE"

# solve the linear programming problem
```

```
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Minimum Profit (z): {model.objective.value()}")
```

```
Optimal Solution
Quantity of product X (x): 5.0
Quantity of product Y (y): 0.0
Minimum Profit (z): 10.0
```

GRAPH PLOTTING

```
# import necesarry libraries
import matplotlib.pyplot as plt
import numpy as np

#define X array
x = np.linspace (0,16,20000)

# Convert constraits to inequalities

y1 = (10 - 2 * x)/3
y2 = (5 - x)/2
y3 = (8 - 3 * x)

# plot constraints

plt.plot(x,y1, label = " 2*x + 3*y >= 10 ")
plt.plot(x,y2, label = "  x + 2*y >= 5 ")
plt.plot(x,y3, label = " 3*x + y >= 8 ")

# Define the feasible region
y3 = np.minimum(y1,y2,y3)
plt.fill_between(x,y3,color="gray", label ="feasible region", alpha = 0.5)
#Define optimal solution
optimal_x = 5
optimal_y = 0
plt.plot(5, 0, "ro", markersize = 8, label = "optimal_solution")

# Define limits
plt.xlim(0,10)
plt.ylim(0,10)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.title("Basic_Resource-Allocation")
plt.legend()
plt.show()
```



/LP/NO1;

## 2 LOAD BALANCING

```
    # importing necessary libraries
from pulp import LpProblem,LpMinimize,LpVariable

# create a linear programming problem
model = LpProblem(name="Load_Balancing", sense=LpMinimize)

# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
```

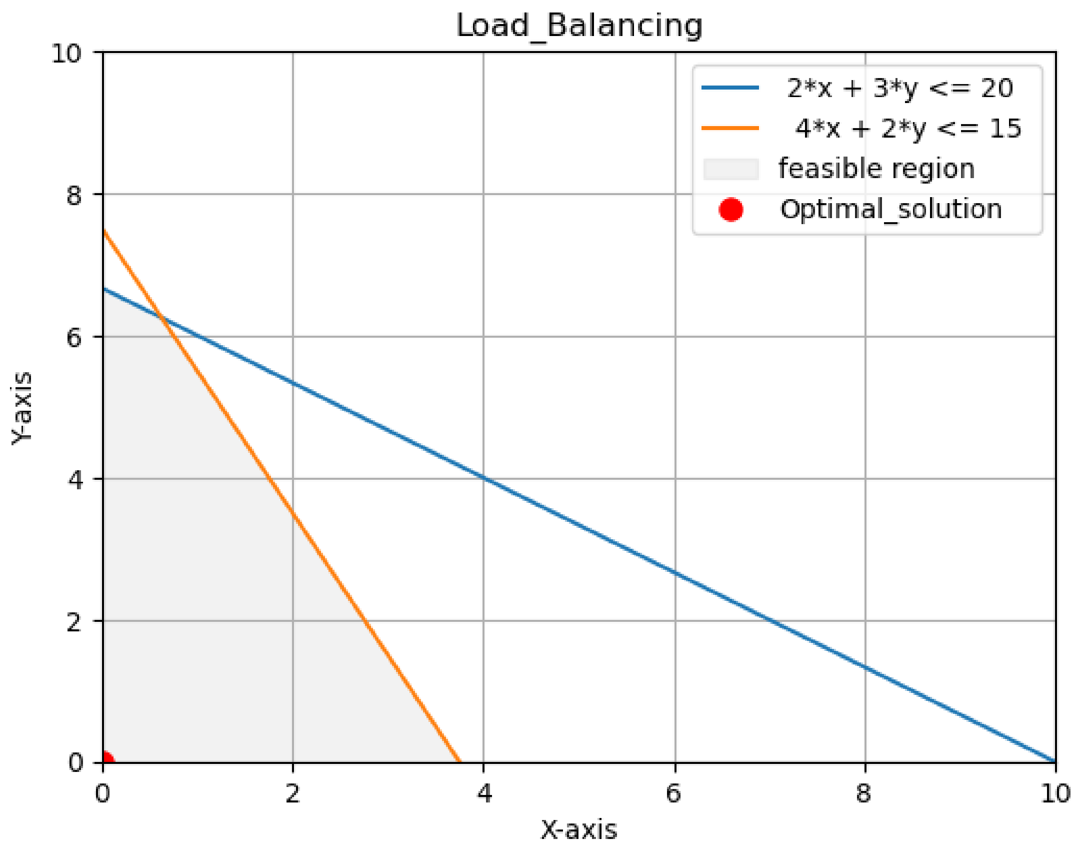
```
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
model  += 5 * x + 4 * y, "Objective"

# Define constraints
model += 2 * x + 3 * y <= 20, "server 1 capacity"
model += 4 * x + 2 * y <= 15, "server 2 capacity"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Minimum Profit (z): {model.objective.value()}")
```

```
Optimal Solution
Quantity of product X (x): 0.0
Quantity of product Y (y): 0.0
Minimum Profit (z): 0.0
```

GRAPH PLOTTING

```
    # import necesarry libraries
import matplotlib.pyplot as plt
import numpy as np

#define X array
x = np.linspace (0,16,500)

# Convert constraits to inequalities

y1 = (20 - 2 * x)/3
y2 = (15 - 4 * x)/2


# plot constraints

plt.plot(x,y1, label = " 2*x + 3*y <= 20 ")
plt.plot(x,y2, label = "  4*x + 2*y <= 15 ")

# Define the feasible region
y3 = np.minimum(y1,y2)
plt.fill_between(x, y3, color="gray", label ="feasible region", alpha = 0.1)
```

```
# Define the optimal solution
optimal_x = 0
optimal_y = 0
plt.plot (0, 0,"ro", markersize = 8, label = "Optimal_solution")

# Define limits
plt.xlim(0,10)
plt.ylim(0,10)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.title("Load_Balancing")
plt.legend()
plt.show()
```



/LP/NO2;

## 3 EFFICIENT ENERGY RESOURCE ALLOCATION

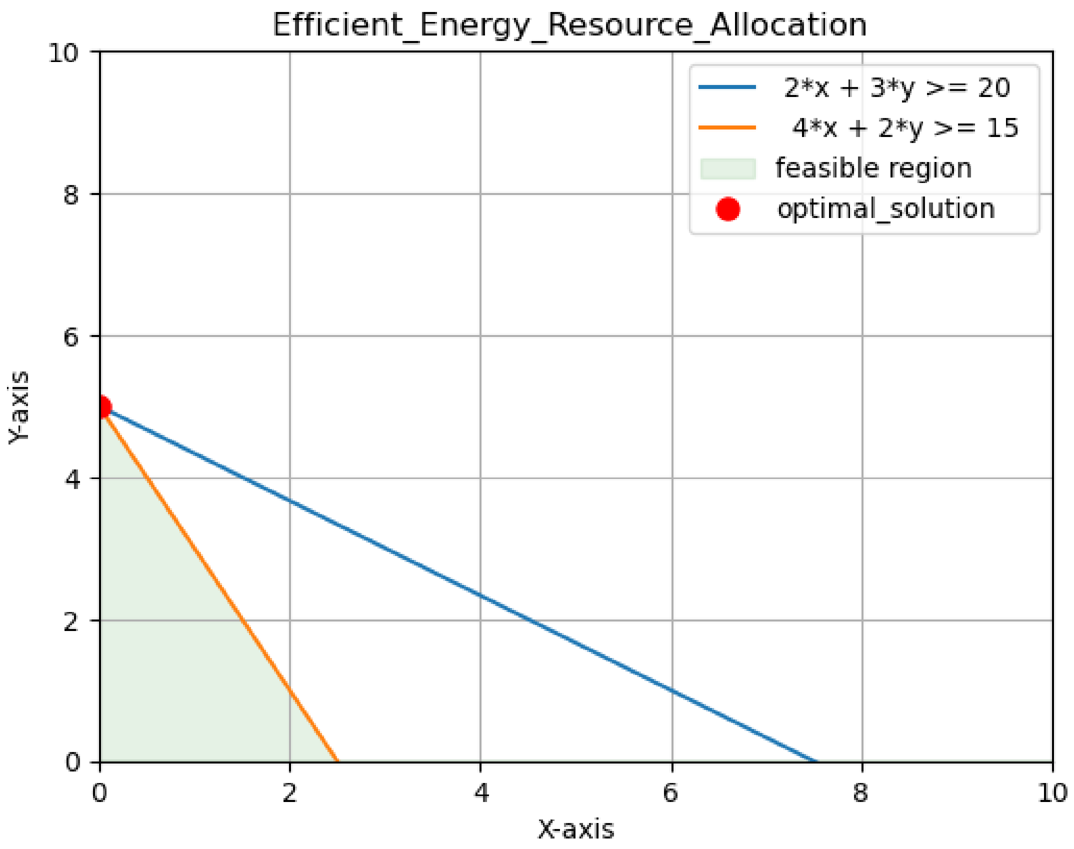
```
    # importing necessary libraries
from pulp import LpProblem,LpMinimize,LpVariable

# create a linear programming problem
model = LpProblem(name="Efficient_Energy_Resource_Allocation", sense=LpMinimize)

# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
model  += 3 * x + 2 * y, "Objective"

# Define constraints
model += 2 * x + 3 * y >= 15, "CPU ALLOCATION"
model += 4 * x + 2 * y >= 10, "MEMORY ALLOCATION"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Minimum Profit (z): {model.objective.value()}")
```

```
Optimal Solution
Quantity of product X (x): 0.0
Quantity of product Y (y): 5.0
Minimum Profit (z): 10.0
```

### GRAPH PLOTTING

```
    # import necesarry libraries

import matplotlib.pyplot as plt
import numpy as np

#define X array
x = np.linspace (0,16,20000)

# Convert constraits to inequalities
```

```
y1 = (15 - 2 * x)/3
y2 = (10 - 4 * x)/2


# plot constraints

plt.plot(x,y1, label = " 2*x + 3*y >= 20 ")
plt.plot(x,y2, label = "  4*x + 2*y >= 15 ")

# Define the feasible region
y3 = np.minimum(y1,y2)
plt.fill_between(x,y3,color="green", label ="feasible region", alpha = 0.1)
# Define the optimal solution
optimal_x = 0
optimal_y = 5
plt.plot(0, 5, "ro", markersize = 8, label ="optimal_solution")
# Define limits
plt.xlim(0,10)
plt.ylim(0,10)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.title("Efficient_Energy_Resource_Allocation ")
plt.legend()
plt.show()
```



/LP/NO3;

## 4 MULTI TENANT RESOURCE SHARING

```
# importing necessary libraries
from pulp import LpProblem,LpMinimize,LpVariable

# create a linear programming problem
model = LpProblem(name="Multi_Tenant_Resource_Sharing", sense=LpMinimize)

# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
model  += 5 * x + 4 * y, "Objective"

# Define constraints
```

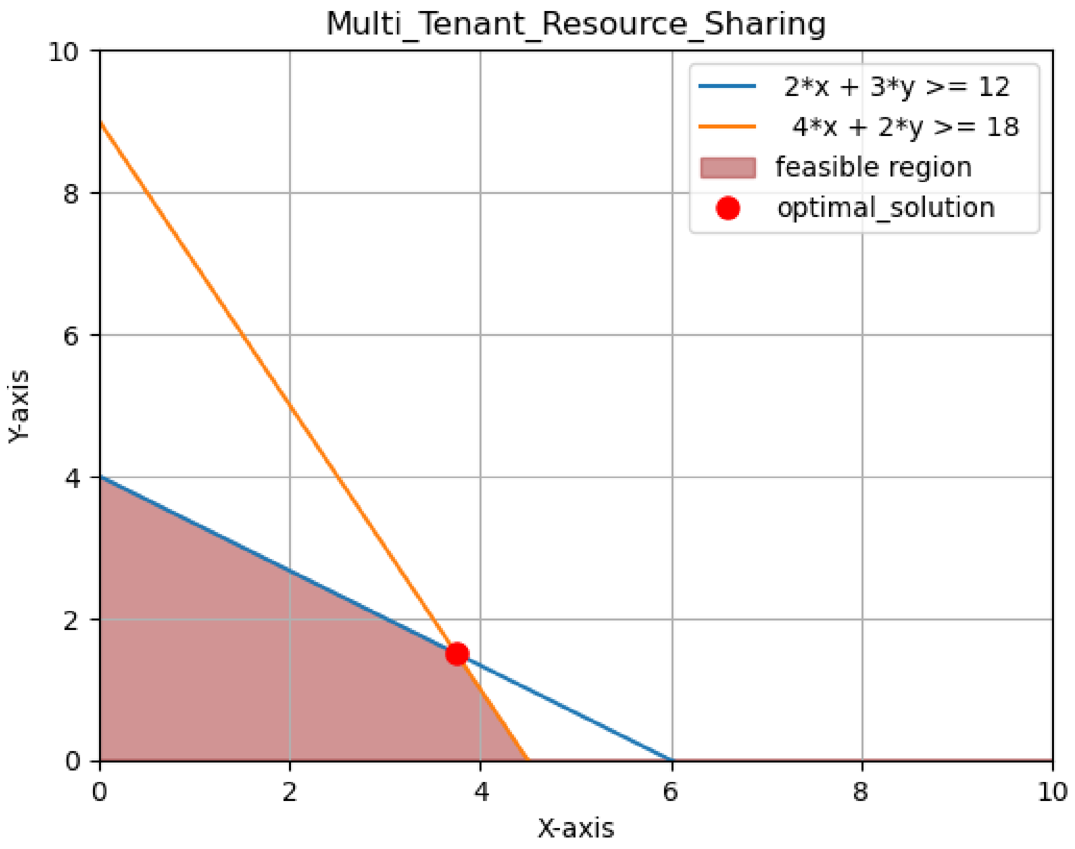
```
model += 2 * x + 3 * y >= 12, "tenant 1"
model += 4 * x + 2 * y >= 18, "tenant 2"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Minimum Profit (z): {model.objective.value()}")
```

```
Optimal Solution
Quantity of product X (x): 3.75
Quantity of product Y (y): 1.5
Minimum Profit (z): 24.75
```

**GRAPH PLOTTING**

```
    # import necesary libraries
import matplotlib.pyplot as plt
import numpy as np

#define X array
x = np.linspace (0,16,200)

# Convert constraits to inequalities

y1 = (12 - 2 * x)/3
y2 = (18 - 4 * x)/2


# plot constraints

plt.plot(x,y1, label = " 2*x + 3*y >= 12 ")
plt.plot(x,y2, label = "  4*x + 2*y >= 18 ")

# Define the feasible region
y3 = np.minimum (y1,y2)
plt.fill_between(x, y3, color="brown", label ="feasible region", alpha = 0.5)
# define optimal solution
optimal_x = 3.75
oOptimal_y = 1.5
plt.plot(3.75, 1.5, "ro", markersize=8, label ="optimal_solution")
# Define limits
plt.xlim(0,10)
plt.ylim(0,10)
```

```
# Define plot
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.title("Multi_Tenant_Resource_Sharing")
plt.legend()
plt.show()
```



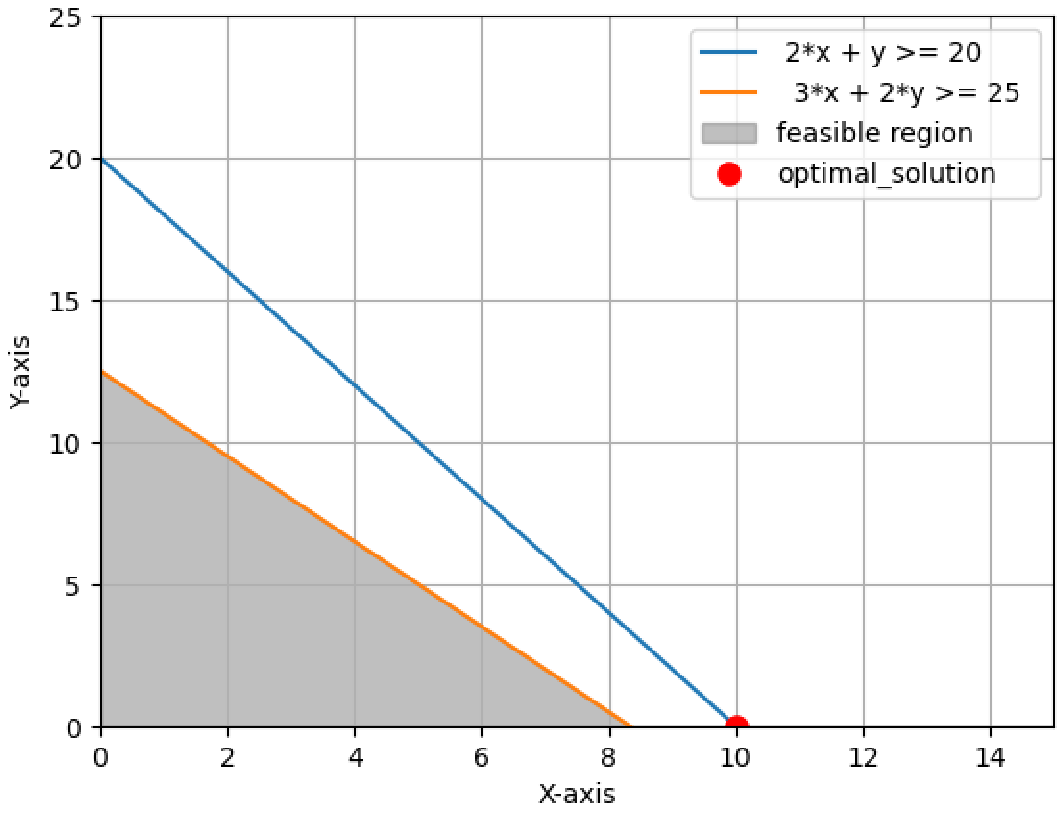
/LP/NO4;

## 5 NO7 DIET OPTIMIZATION

```
    from pulp import *

model = pulp.LpProblem('Deit_Optimization', sense=LpMinimize)

# get solver
solver = getSolver('PULP_CBC_CMD')
# importing necessary libraries
# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
model  += 3 * x + 2 * y, "Objective"

# Define constraints
model += 2 * x +  y >= 20, "Proteins"
model += 3 * x + 2 * y >= 25, "Vitamins"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Minimum Profit (z): {model.objective.value()}")

Optimal Solution
Quantity of product X (x): 10.0
Quantity of product Y (y): 0.0
Minimum Profit (z): 30.0
```

**GRAPH PLOTTING**

```
    # import necesarry libraries
import matplotlib.pyplot as plt
import numpy as np

#define X array
y1 = np.linspace (0,15,400)
y2= np.linspace (0,15,400)

# Convert constraits to inequalities

y1 = (20 - 2 * x)
y2 = (25 - 3 * x)/2


# plot constraints

plt.plot(x,y1, label = " 2*x + y >= 20 ")
plt.plot(x,y2, label = "  3*x + 2*y >= 25 ")

# Define the feasible region
y3 = np.minimum(y1,y2)
plt.fill_between(x,y3,color="gray", label ="feasible region", alpha = 0.5)
# define the optimal solution
optimal_x = 10
optimal_y = 0
plt.plot(10, 0, "ro", markersize = 8, label ="optimal_solution")



# Define limits
plt.xlim(0,15)
plt.ylim(0,25)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.legend()
plt.show()
```



/LP/NO5;

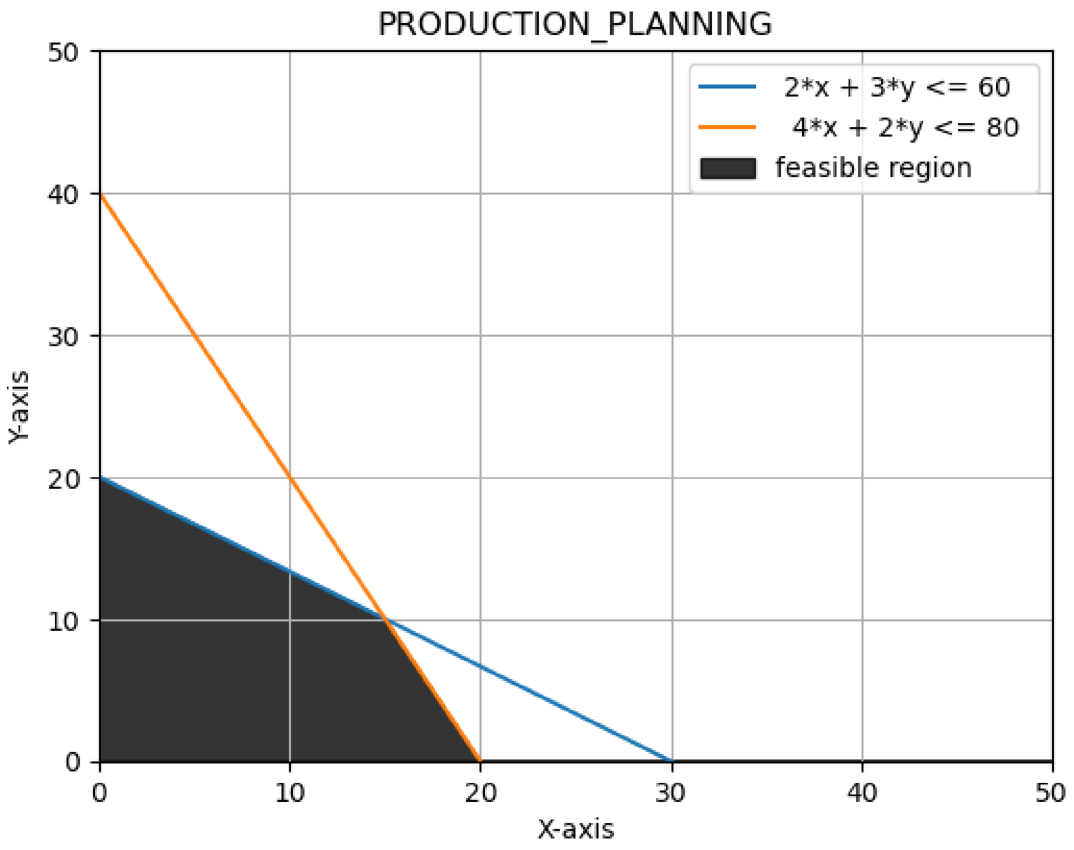
## 6 PRODUCTION PLANNING

```
    from pulp import *

model = pulp.LpProblem('Production_Planning', sense=LpMaximize)

# get solver
solver = getSolver('PULP_CBC_CMD')
# importing necessary libraries


# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y


#Define the objective function
```

```
model  += 5 * x + 3 * y, "Objective"

# Define constraints
model += 2 * x + 3 * y <= 60, "labor"
model += 4 * x + 2 * y <= 80, "raw materials"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Maximum Profit (z): {model.objective.value()}")

Optimal Solution
Quantity of product X (x): 15.0
Quantity of product Y (y): 10.0
Maximum Profit (z): 105.0
```

## GRAPH PLOTTING

```
    # import necesarry libraries
import matplotlib.pyplot as plt
import numpy as np

#define X array
x = np.linspace (0,100,20000)

# Convert constraits to inequalities

y1 = (60 - 2 * x)/3
y2 = (80 - 4 * x)/2


# plot constraints

plt.plot(x,y1, label = " 2*x + 3*y <= 60 ")
plt.plot(x,y2, label = "  4*x + 2*y <= 80 ")

# Define the feasible region
y3 = np.minimum.reduce([y1,y2])
plt.fill_between(x,y3,color="black", label ="feasible region", alpha = 0.8)


# Define limits
plt.xlim(0,50)
plt.ylim(0,50)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid("true")
plt.title("PRODUCTION_PLANNING")
plt.legend()
plt.show()
```



/LP/NO6;

## 7 production planning in manufacturing

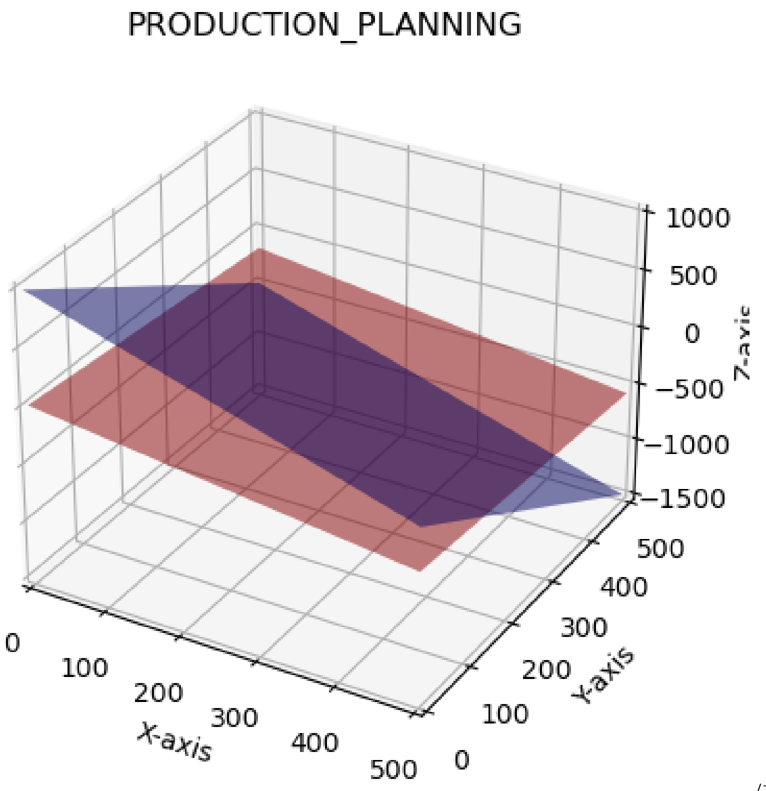
```
    # importing necessary libraries
from pulp import LpProblem,LpMinimize,LpVariable

# create a linear programming problem
model = LpProblem(name="Production_Planning_In_Manufacturing", sense=LpMinimize)
```

```
# Define decision variables
x = LpVariable(name="x", lowBound=0) # Quantity of product X
y = LpVariable(name="y", lowBound=0) # Quantity of product Y
z = LpVariable(name="z", lowBound=0) # Quantity of product Z


#Define the objective function
model  += 5 * x + 3 * y + 4 * z, "Objective"

# Define constraints
model += 2 * x + 3 * y + z <= 1000, "RAW MATERIAL"
model +=  4 * x + 2 * y  + 5 * z <= 120, "LABOR"
model += x >= 200, "DEMAND_1"
model += y >= 300, "DEMAND_2"
model += z >= 150, "DEMAND_3"

# solve the linear programming problem
model.solve()

# Display the results
print("Optimal Solution")
print(f"Quantity of product X (x): {x.varValue}")
print(f"Quantity of product Y (y): {y.varValue}")
print(f"Quantity of product Z (z): {z.varValue}")
print(f"Minimum Profit (Z): {model.objective.value()}")

Optimal Solution
Quantity of product X (x): 200.0
Quantity of product Y (y): 300.0
Quantity of product Z (z): 0.0
Minimum Profit (Z): 1900.0
```

GRAPH PLOTTING

```
    # import necesarryr4 libraries
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
# Create a 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection = "3d")

#define X,Y,Z arrays
x = np.linspace (0,500,100)
y = np.linspace (0,500,100)
x,y  = np.meshgrid (x,y)
```

```
#Calculating corresponding constraints
z1=(1000 - 2 * x - 3 * y )
z2=(120 - 4 * x - 2 * y )/5

# plot constraints

ax.plot_surface(x,y,z1,alpha = 0.5, color="blue", rstride = 100, cstride = 100 )
ax.plot_surface(x,y,z2, alpha = 0.5, color="red", rstride = 100, cstride = 100 )

# Define limits
plt.xlim(0,500)
plt.ylim(0,500)


ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.grid("true")
plt.title("PRODUCTION_PLANNING")

plt.show()
```


PRODUCTION_PLANNING
X-axis
Y-axis
Z-axis
0
100
200
300
400
500
0
100
200
300
400
500
1000
500
0
−500
−1000
−1500

/LP/NO7;