LINEAR PROGRAMMING INDIVIDUAL COURSE WORK

NYALWA KENETH

February 2024

1 MOUNTAIN OF THE MOON UNIVERSITY

2 FACULTY OF SCIENCE TECHNOLOGY AND INNOVATION

3 DEPARTMENT OF COMPUTER SCIENCE

4 LECTURERS NAME : MR. OCEN SAMUEL

5 COURSE CODE: BCS 1202

6 COURSE NAME : LINEAR PROGRAMMING

7 YEAR ONE

8 SEMESTER TWO

8.0.1 number 1

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

#create a linear programming problem
model= LpProblem(name="basic_Resource_Allocation_Optimazation", 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 += 4 * x + 5 * y, "objective"

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

#solving 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("Minimum profit (z):", model.objective.value())
```

```
#importing numpy as np
import numpy as np

# Importing matplotlib.pyplot
import matplotlib.pyplot as plt

#Define the constraints
x = np.linspace(0,10,300)
y1= (10 - 2 * x)/3
y2= (5 - x)/2
y3= (8 - 3 * x)

#plot the constraints
plt.plot(x,y1, label = "2x + 3y >= 10")
```

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

#define the feasible region
# plt.fill_between(x,np.minimum(y1, y2, y3), color="green", alpha=0.5)
x=[0,2.2,2.67,0]
y=[2.5,1.4,0,0]
plt.fill(x,y,color='yellow', alpha= 0.5 , label="unwanted region")
#add labels and titles
optimal_x=2.0
optimal_y=2.0
plt.plot(2,2, "ro", markersize=10, label="optimal_solutiion")
plt.xlabel("x")
plt.ylabel("y")
plt.title("basic resource allocation graph")
plt.legend()
plt.grid(True)
plt.xlim(0,10)
plt.ylim(0,10)
plt.show
Optimal solution
Quantity of product X (x): 2.0
Quantity of product Y (y): 2.0
Minimum profit (z): 18.0
```

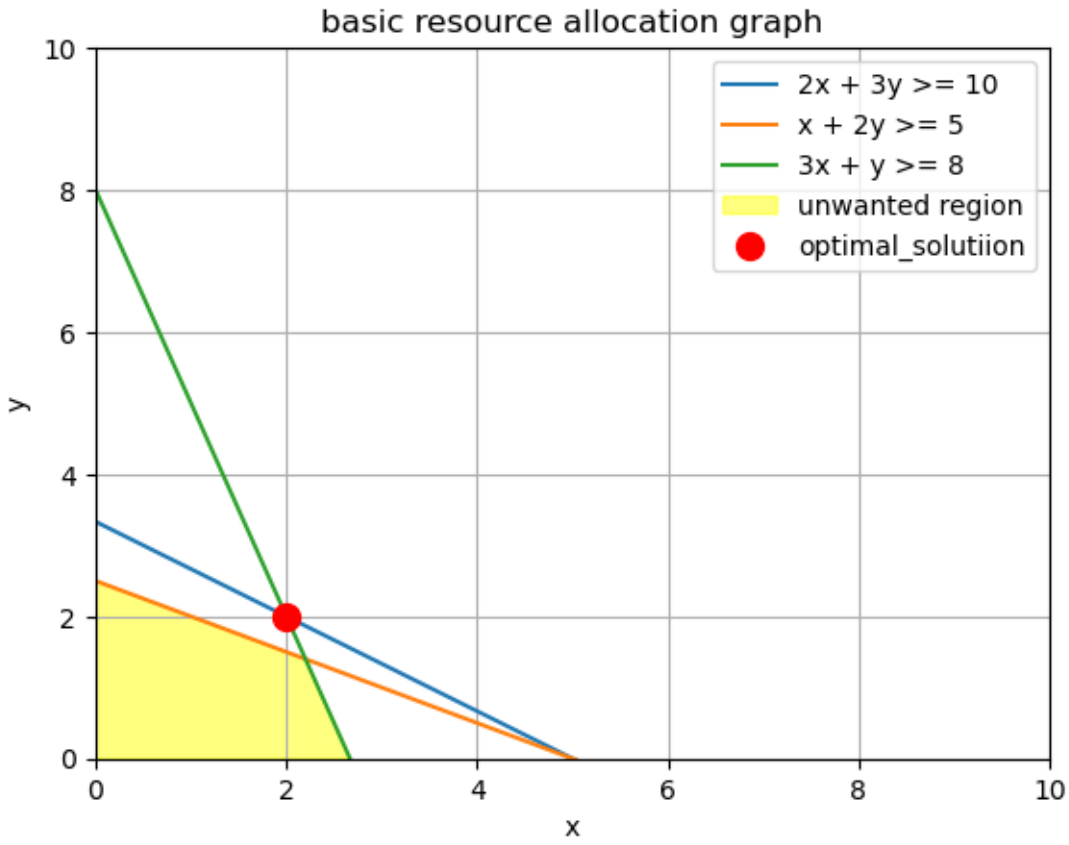

Figure 1: graph 1

8.0.2 number 2

```
    #IMPORTING NECESSARY LIBRARIES
from pulp import LpProblem, LpVariable, LpMinimize

#creating a linear programming problem
model = LpProblem(name="minimize_the_overall_responce_time", sense=LpMinimize)

#define decision variables
x= LpVariable(name = "X", lowBound=0) #capacity of server1
y= LpVariable(name = "Y", lowBound=0) #capaaccity of sever2

#define the objective function
model += 5 * x + 4 * y, "obctive"

#define the constrains
model += 2 * x + 3 * y <= 20, "server1"
model += 4 * x + 2 * y <= 15, "server2"

#solving linear programming problem
model.solve()

#display the results
print("Optimal solution")
print(f"Quantity of server1 X (x):", x.varValue)
print(f"Quantity of server2 Y (y):", y.varValue)
print("Minimum profit (z):", model.objective.value())



import numpy as np
import matplotlib.pyplot as plt

# Define the constraints
x_values = np.linspace(0, 10, 400)
y1_values = (20 - 2 * x_values) / 3  # Constraint 1: 2x + 3y <= 20
y2_values = (15 - 4 * x_values) / 2  # Constraint 2: 4x + 2y <= 15

# Plot the constraints
```

```
plt.plot(x_values, y1_values, label=r'$2x + 3y \leq 20$')
plt.plot(x_values, y2_values, label=r'$4x + 2y \leq 15$')

# Plot the feasible region (shaded)
plt.fill_between(x_values, np.minimum(y1_values, y2_values), where=(y1_values >= 0) & (y2_va

# Plot the optimal solution point
x_optimal = 3  # Replace with the optimal value of x
y_optimal = 4  # Replace with the optimal value of y
plt.plot(x_optimal, y_optimal, 'ro', label='Optimal Solution')

# Add labels and legend
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title('Load balancing graph')
plt.legend()
plt.grid(True)
plt.xlim(0, 10)
plt.ylim(0, 10)
plt.show()
```

```
Optimal solution
Quantity of server1 X (x): 0.0
Quantity of server2 Y (y): 0.0
Minimum profit (z): 0.0
```

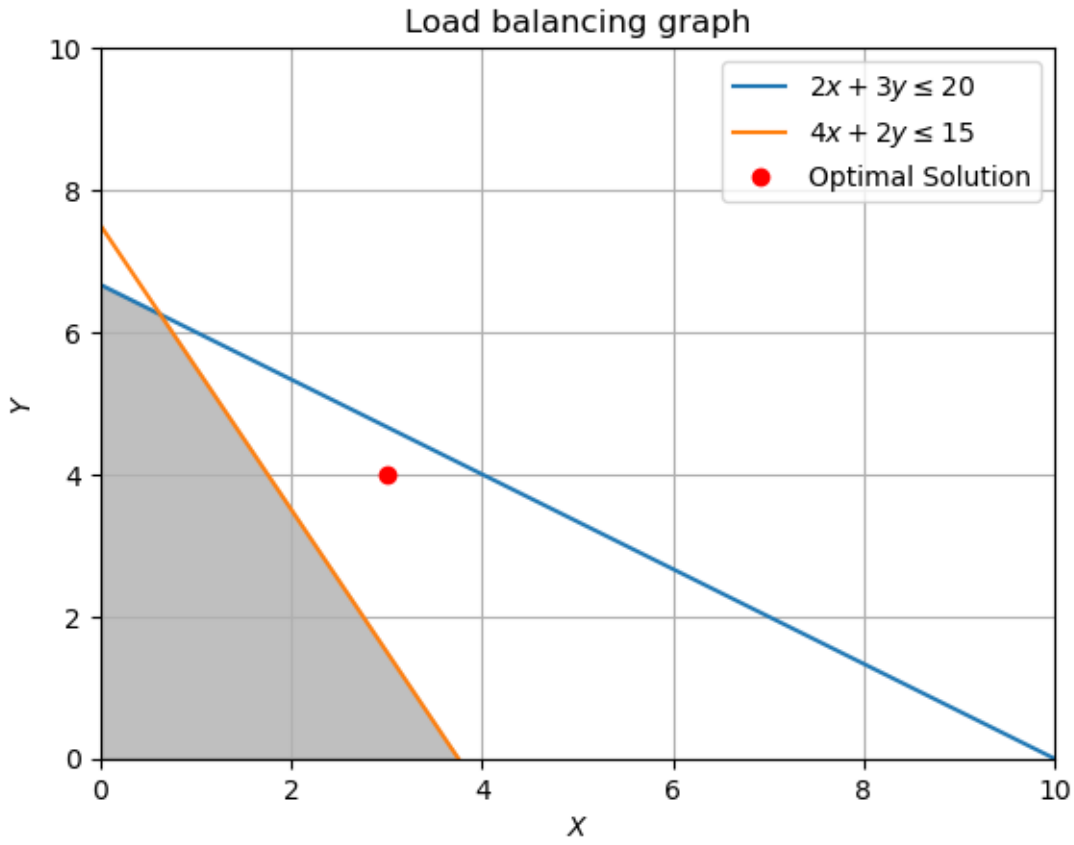

Figure 2: graph 2

8.0.3 Number 3

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

#create a linear programming plroblem
model = LpProblem(name="Energy_Effient_resource_allocation", sense=LpMinimize)


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

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

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

#solving 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("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



#import numpy as np
import matplotlib.pyplot as plt

# Define the constraint equations
x_vals = np.linspace(0, 10, 400)
constraint1 = (15 - 2*x_vals) / 3
constraint2 = (10 - 4*x_vals) / 2

# Plot the constraints
plt.plot(x_vals, constraint1, label='2x + 3y >= 15')
```

```
plt.plot(x_vals, constraint2, label='4x + 2y >= 10')
plt.fill_between(x_vals, np.maximum(constraint1, constraint2), color='gray', alpha=0.5)

# Define the objective function
z = (45 - 3*x_vals) / 2  # Assuming the objective value is 45

# Plot the objective function
plt.plot(x_vals, z, label='3x + 2y = 45', linestyle='--')

# Add labels and legend
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Energy efficient and resource allocation')
plt.legend()
plt.grid(True)

# Highlight the optimal point
optimal_x = 5  # Assuming the optimal value of x is 5
optimal_y = (45 - 3*optimal_x) / 2  # Calculating y using the objective function
plt.scatter(optimal_x, optimal_y, color='red', label='Optimal Solution')

# Show plot
plt.show()
```

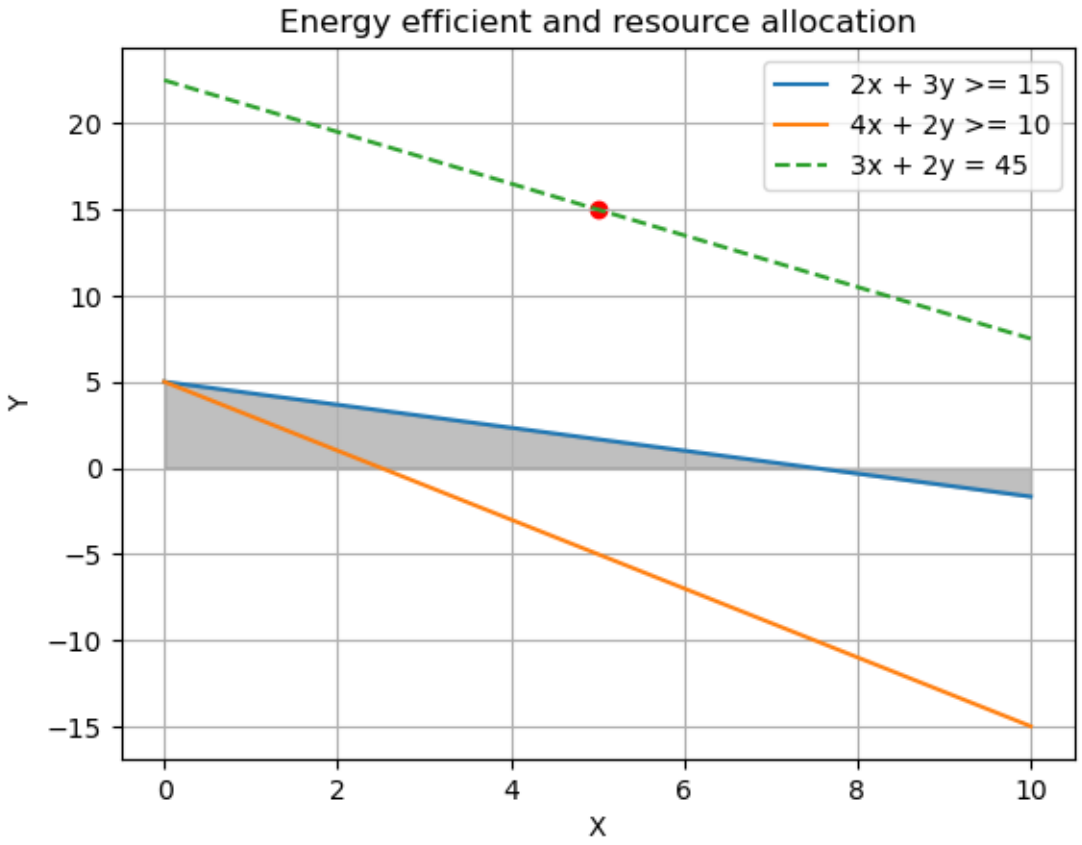

Figure 3: graph 3

8.0.4 4

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

# Define the  linear programming problem
model = LpProblem(name="multiple_tenant_resource_sharing", sense = LpMinimize)

#define variables
x = LpVariable("x", 0)
y = LpVariable("y", 0)


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

# define constraints
model += 2 * x + 3 * y >= 12, "tenant1"
model += 4 * x + 2 * y >= 18, "tenant2"

#solving Linear programming
model.solve()

#desplay the results
print("optimal solution: ")
print(f"Quantity of tenant1 X(x) :", x.varValue)
print(f"Quantity of tenant2 Y(y):", y.varValue)
print("Minimum of product (Z):",model.objective.value())

optimal solution:
Quantity of tenant1 X(x) : 3.75
Quantity of tenant2 Y(y): 1.5
Minimum of product (Z): 24.75

import numpy as np
import matplotlib.pyplot as plt

# Define the constraints as equations
x_values = np.linspace(0, 10, 400)
y_values_1 = (12 - 2*x_values) / 3  # From the constraint 2*x + 3*y >= 12
y_values_2 = (18 - 4*x_values) / 2  # From the constraint 4*x + 2*y >= 18

# Plot the constraints
plt.plot(x_values, y_values_1, label=r"$2x + 3y \geq 12$")
plt.plot(x_values, y_values_2, label=r"$4x + 2y \geq 18$")
```

```
# Plot non-negativity constraints
plt.fill_between(x_values, 0, 100, where=(x_values >= 0), color='gray', alpha=0.2)
plt.fill_betweenx(y_values_1, 0, 100, where=(y_values_1 >= 0), color='gray', alpha=0.2)
plt.fill_betweenx(y_values_2, 0, 100, where=(y_values_2 >= 0), color='gray', alpha=0.2)

# Plot the feasible region
plt.fill_between(x_values, np.maximum(y_values_1, y_values_2), 100, color='blue', alpha=0.3

# Plot the optimal solution
plt.scatter([2.4], [3.6], color='red', label="Optimal Solution (x=2.4, y=3.6)")

# Label the axes and add legend
plt.xlabel("x")
plt.ylabel("y")
plt.title("Muliti tenant resource sharing graph")
plt.legend()

# Set x and y axis limits
plt.xlim(0, 10)
plt.ylim(0, 10)

# Show plot
plt.grid(True)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.show()
```

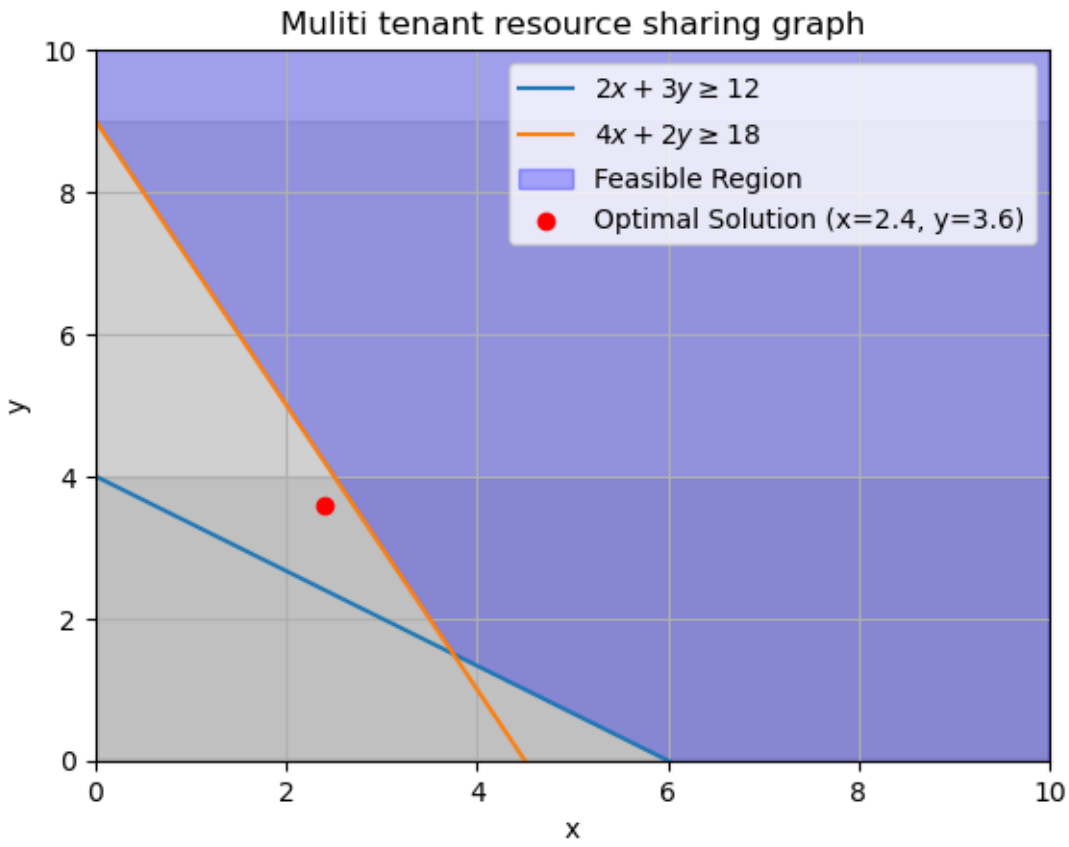

Figure 4: graph 4

8.0.5 number 5

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

# Define the  linear programming problem
model = LpProblem(name="production_cost_optimization", sense = LpMinimize)

#define variables
x1 = LpVariable("x1", 0)
x2 = LpVariable("x2", 0)
x3 = LpVariable("x3", 0)

#objective function
model += 5 * x1 + 3 * x2 + 4 * x3, "objective"

# define constraints
model += 2 * x1 + 3 * x2 + x3 <=1000, "raw_materials"
model += 4 * x1 + 2 * x2 + 5 * x3 <= 120, "Labor_hours"
model += x1 >=200, "minimum1"
model += x2 >=300, "minimum2"
model += x3 >=150 ,"minimum3"

#solving Linear programming
model.solve()

#desplay the results
print("optimal solution: ")
print(f"quantity of product X1(x1):",x1.varValue)
print(f"quantity of product X2 (x2):", x2.varValue)
print(f"quantity of product X3(x3):",x3.varValue)
print("Minimum of product (Z):",model.objective.value())


optimal solution:
quantity of product X1(x1): 200.0
quantity of product X2 (x2): 300.0
quantity of product X3(x3): 0.0
Minimum of product (Z): 1900.0

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define the range of values for x1, x2, and x3
x1_values = np.linspace(0, 500, 100)  # Adjust the range as needed
```

```
x2_values = np.linspace(0, 500, 100)  # Adjust the range as needed
x3_values = np.linspace(0, 500, 100)  # Adjust the range as needed

# Create a meshgrid of x1, x2, and x3 values
X1, X2, X3 = np.meshgrid(x1_values, x2_values, x3_values)

# Calculate the objective function values for each combination of x1, x2, and x3
Z = 5 * X1 + 3 * X2 + 4 * X3

# Flatten X1, X2, and Z arrays
X1_flat = X1.flatten()
X2_flat = X2.flatten()
Z_flat = Z.flatten()

# Plot the 3D surface
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_trisurf(X1_flat, X2_flat, Z_flat, cmap='viridis')

# Plot x1, x2, and x3 variables against the objective function
ax.scatter(x1_values, np.zeros_like(x1_values), 5 * x1_values, color='r', label='x1')
ax.scatter(np.zeros_like(x2_values), x2_values, 3 * x2_values, color='g', label='x2')
ax.scatter(np.zeros_like(x3_values), np.zeros_like(x3_values), 4 * x3_values, color='b', lab

# Set labels and title
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('Objective Function Value')
ax.set_title('Objective Function Surface')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

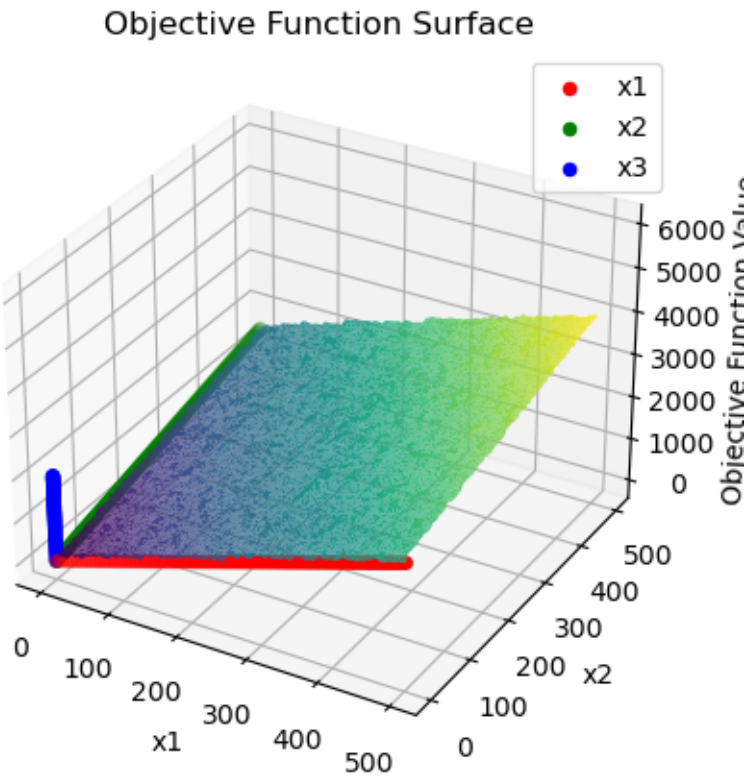

Figure 5: graph 5

8.0.6 number 6

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

#create a linear programming minimization problem
model= LpProblem(name="Maximize_the_return_on_investment", sense=LpMaximize)

#defining the  objective variables
x1 = LpVariable(name=("x1"), lowBound=0) #investment in stock A
x2 = LpVariable(name=("x2"), lowBound=0) #investment in stock B
x3 = LpVariable(name=("x3"), lowBound=0) #investment in stock C

#defining the objective variable
model += 0.08 * x1 + 0.1 * x2 + 0.12 * x3

#defining the constraints
model += 2 * x1 + 3 * x2 + x3 <= 10000, "budget constraint (maximum budget for investment)"
model += x2 >=2000,"minimum investment1"
model += x2 >=1500, "minimum investment2"
model += x3 >=1000, "minimum investment3"

#solve the problem
model.solve()

#display the result
print("optimal value:")
print(f"optimal value (x1): {x1.varValue}")
print(f"optimal value (x2): {x2.varValue}")
print(f"optimal value (x3): {x3.varValue}")
print(f"maximum_return_on_investment (Z): {model.objective.value()}")

optimal value:
optimal value (x1): 0.0
optimal value (x2): 2000.0
optimal value (x3): 4000.0
maximum_return_on_investment (Z): 680.0

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define the constraints
def constraint1(x1):
    return (10 - 2*x1)
```

```
def constraint2(x1):
    return ((10 - x1) / 2)

# Define the feasible region
x1_values = np.linspace(0, 10, 400)
y1_values = constraint1(x1_values)
y2_values = constraint2(x1_values)

# Now let's plot the 3D graph
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Define the objective function Z = x1 + x2
X1, X2 = np.meshgrid(x1_values, x1_values)
Z = X1 + X2

# Plot the objective function surface
ax.plot_surface(X1, X2, Z, alpha=0.5, cmap='viridis')

# Plot the constraints in 3D
ax.plot(x1_values, constraint1(x1_values), zs=0, zdir='y', label=r'$2x_1 + x_2 \leq 10$')
ax.plot(x1_values, constraint2(x1_values), zs=0, zdir='y', label=r'$x_1 + 2x_2 \leq 10$')

# Plot the intersection point in 3D
ax.scatter(intersection_x, intersection_y, intersection_x + intersection_y, color='red', lab


# Add labels and legend for 3D plot
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$Z$')
plt.title('Financial portfolio optimisation graph')
plt.legend()
```

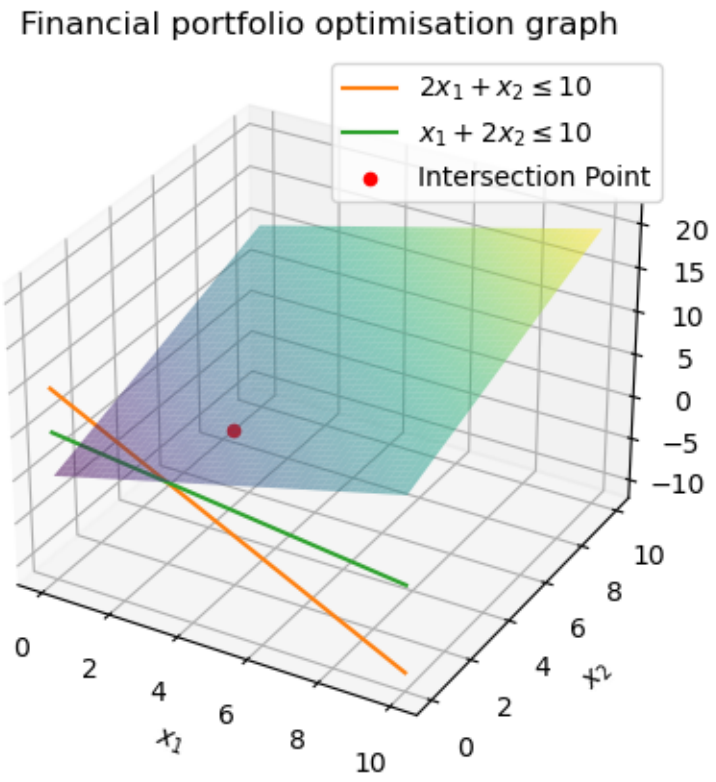

Figure 6: graph 6

8.0.7 number 7

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

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

#define variables
x1 = LpVariable("x1", 0)
x2 = LpVariable("x2", 0)

#objective function
model += 3 * x1 + 2 *x2, "objective"

# define constraints
model += 2 * x1 +  x2 >= 20, "proteins_grams"
model += 3 * x1 + 2 * x2 >= 25, "vitamins_units"

#solving Linear programming
model.solve()

#desplay the results
print("optimal solution: ")
print(f"serving of food item1 X1(x1):",x1.varValue)
print(f"serving  of food item2 X2 (x2):", x2.varValue)
print("Minimum of product (Z):",model.objective.value())

optimal solution:
serving of food item1 X1(x1): 10.0
serving  of food item2 X2 (x2): 0.0
Minimum of product (Z): 30.0



import numpy as np
import matplotlib.pyplot as plt

# Define the constraints as equations
x1_values = np.linspace(0, 15, 400)
x2_values_1 = (20 - 2*x1_values)  # From the constraint 2*x1 + x2 >= 20
x2_values_2 = (25 - 3*x1_values) / 2  # From the constraint 3*x1 + 2*x2 >= 25

# Plot the constraints
plt.plot(x1_values, x2_values_1, label=r"$2x_1 + x_2 \geq 20$")
plt.plot(x1_values, x2_values_2, label=r"$3x_1 + 2x_2 \geq 25$")
```

```
# Plot non-negativity constraints
plt.fill_between(x1_values, 0, 100, where=(x1_values >= 0), color='gray', alpha=0.2)
plt.fill_betweenx(x2_values_1, 0, 100, where=(x2_values_1 >= 0), color='gray', alpha=0.2)
plt.fill_betweenx(x2_values_2, 0, 100, where=(x2_values_2 >= 0), color='gray', alpha=0.2)

# Plot the feasible region
plt.fill_between(x1_values, np.maximum(x2_values_1, x2_values_2), 100, color='blue', alpha=(

# Plot the optimal solution
plt.scatter([3.75], [8.75], color='red', label="Optimal Solution (x1=3.75, x2=8.75)")

# Label the axes and add legend
plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Diet optmissation")
plt.legend()

# Set x and y axis limits
plt.xlim(0, 15)
plt.ylim(0, 15)

# Show plot
plt.grid(True)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.show()
```

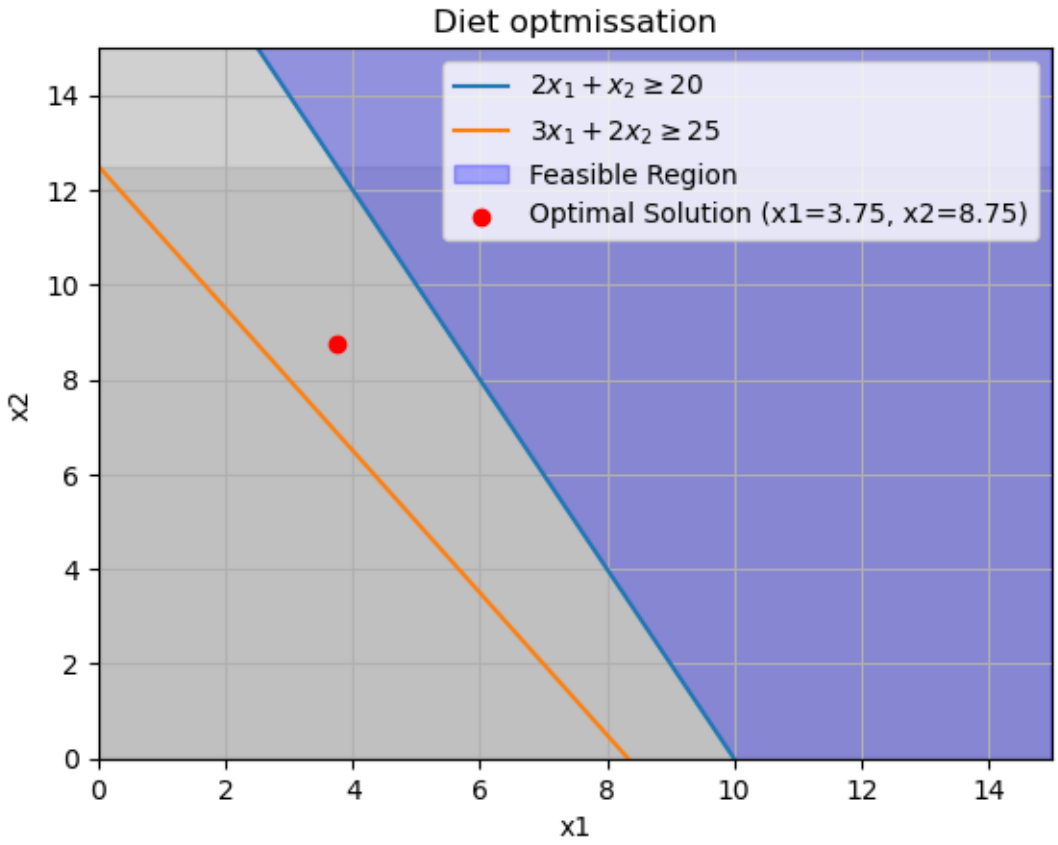

Figure 7: graph 7

8.0.8 number 8

```
    #import necessary libralies
from  pulp import LpProblem, LpVariable, LpMaximize

#create a linear programming problem
model = LpProblem(name= "production_planning", sense = LpMaximize)

#define variables
x1 =LpVariable("x1", 0)
x2 =LpVariable("x2", 0)

#define the objective function
model += 5 * x1 + 3 * x2, "objective function"

# define constraints
model += 2 * x1 + 3 * x2 <= 60, "labor_hours"
model += 4 * x1 + 2 * x2 <= 80, "raw_material_units"

#solving Linear programming
model.solve()

#desplay the results
print("optimal solution: ")
print(f"Quantity of product1 to produce X1(x1):",x1.varValue)
print(f"Quantity of product2 to produce X2 (x2):", x2.varValue)
print("Maximize of product (Z):",model.objective.value())


optimal solution:
Quantity of product1 to produce X1(x1): 15.0
Quantity of product2 to produce X2 (x2): 10.0
Maximize of product (Z): 105.0


import numpy as np
import matplotlib.pyplot as plt
from pulp import LpProblem, LpVariable, LpMaximize, value

# Create a linear programming problem
model = LpProblem(name="production_planning", sense=LpMaximize)

# Define variables
x1 = LpVariable("x1", lowBound=0)  # Setting lower bound explicitly
x2 = LpVariable("x2", lowBound=0)
```

```
# Define the objective function
model += 5 * x1 + 3 * x2, "objective function"

# Define constraints
model += 2 * x1 + 3 * x2 <= 60, "labor_hours"
model += 4 * x1 + 2 * x2 <= 80, "raw_material_units"

# Solving Linear programming
model.solve()

# Extract optimal values
opt_x1 = value(x1)
opt_x2 = value(x2)

# Define the constraints
x = np.linspace(0, 30, 400)
y1 = (60 - 2*x)/3
y2 = (80 - 4*x)/2

# Plot the constraints
plt.plot(x, y1, label=r'$2x_1 + 3x_2 \leq 60$', color='green')
plt.plot(x, y2, label=r'$4x_1 + 2x_2 \leq 80$', color='blue')

# Plot the feasible region
plt.fill_between(x, np.minimum(y1, y2), 0, where=(x >= 0) & (y1 >= 0) & (y2 >= 0), color='sk

# Define the objective function
z = lambda x1, x2: 5*x1 + 3*x2

# Generate grid for objective function plot
X1, X2 = np.meshgrid(np.linspace(0, 30, 100), np.linspace(0, 30, 100))
Z = z(X1, X2)

# Plot the objective function contours
plt.contour(X1, X2, Z, levels=np.arange(0, 201, 10), cmap='viridis', alpha=0.7, linestyles='
# Plot the optimal solution
plt.scatter(opt_x1, opt_x2, color='red', label='Optimal Solution')
# Labels and legend
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.title('production planning graph')
plt.legend()
# Show plot
plt.grid(True)
plt.axis([0, 30, 0, 30])
plt.show()
```

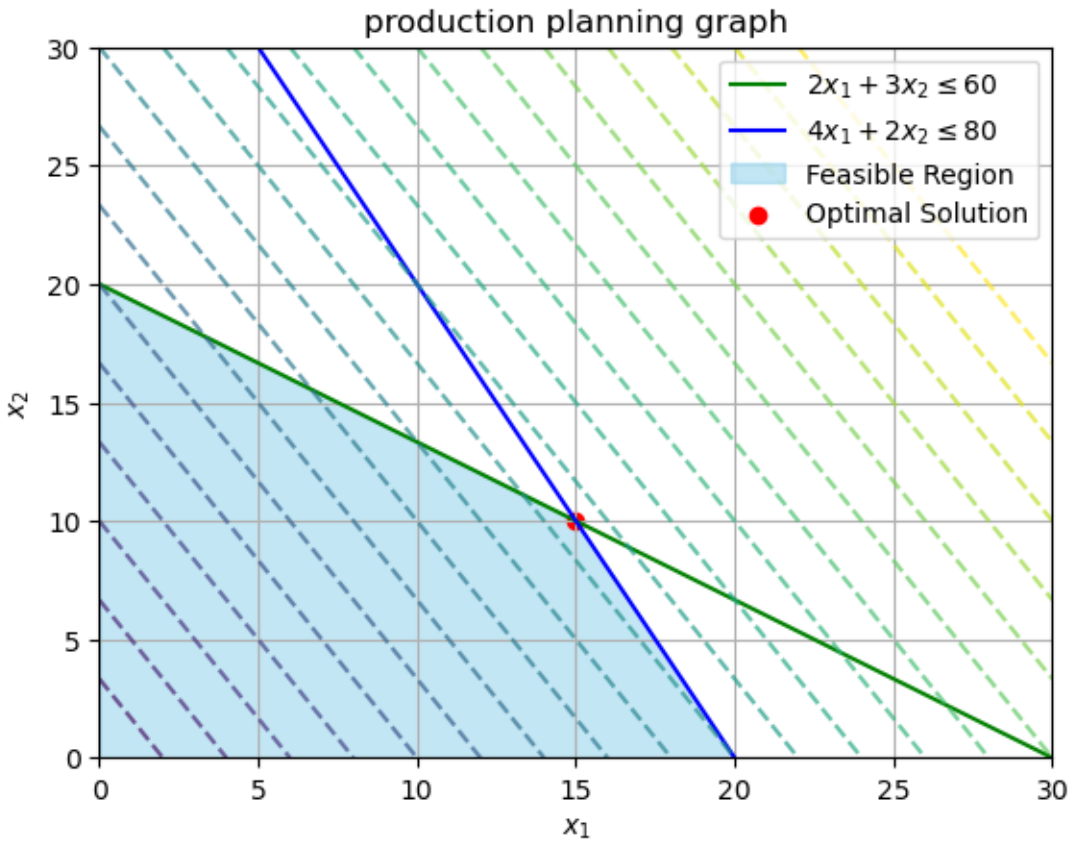

Figure 8: graph 8