**MOUNTAINS OF THE MOON UNIVERSITY**

**FACULTY OF SCIENCE TECHNOLOGY**

**AND INNOVATION**

**DEPARTMENT OF COMPUTER SCIENCE**

**Coursework**

**NAME: MUHINDO JOSEPH**

**REGISTRATION NUMBER: 230270/U/MMU/BCS/01139**

**COURSE CODE: BCS 1201**

**LECTURER'S NAME: Mr. Samuel Ocen**

# 1 Resource Allocation in Cloud Computing

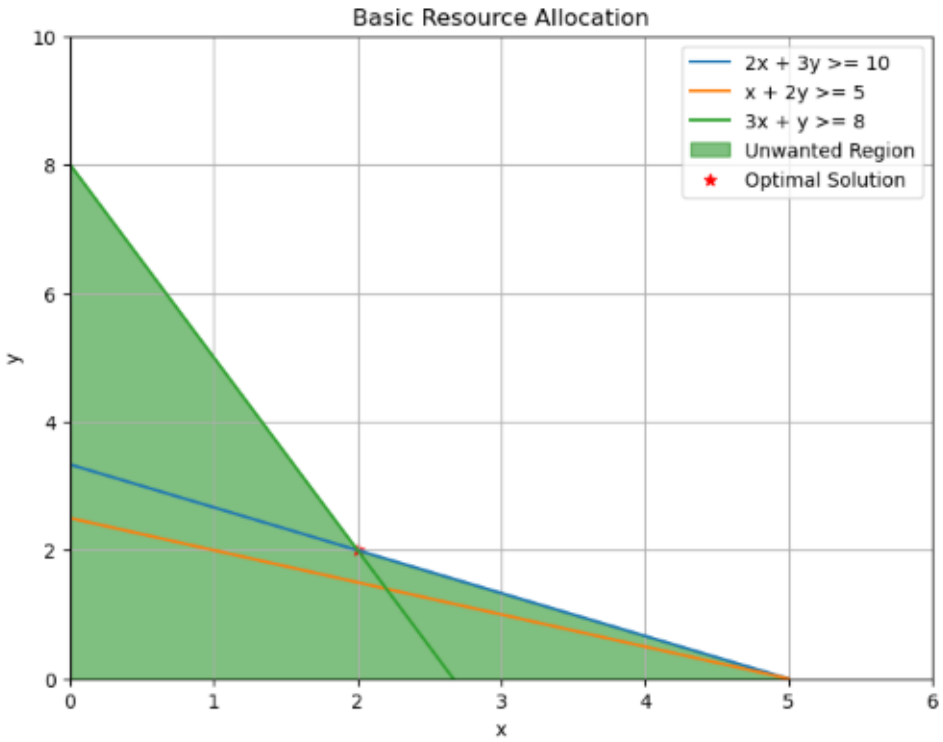
## 1.1 Number One: Basic Resource Allocation

```
#importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import linprog

# Define the objective function coefficients
c = [4, 5]

# Coefficients of the inequality constraints
A = np.array([
[-2, -3],  # CPU constraint
[-1, -2],  # Memory constraint
[-3, -1],  # Storage constraint
])

# Right-hand side values of the inequality constraints
b = [-10, -5, -8]

# Solve linear programming problem
result = linprog(c, A_ub=A, b_ub=b)

# Display the results
print("Optimal values:")
print("x =", result.x[0])
print("y =", result.x[1])
print("Optimal objective function value (z) =", result.fun)

# Plotting the feasible region
x_values = np.linspace(0, 5, 100)
y1_values = (10 - 2*x_values) / 3
y2_values = (5 - x_values) / 2
y3_values = 8 - 3*x_values

plt.figure(figsize=(8, 6))
plt.plot(x_values, y1_values, label='2x + 3y >= 10')
plt.plot(x_values, y2_values, label='x + 2y >= 5')
plt.plot(x_values, y3_values, label='3x + y >= 8')

plt.fill_between(x_values, np.maximum(np.maximum(y1_values, y2_values), y3_values),
plt.scatter(result.x[0], result.x[1], color='red', marker='*', label='Optimal Soluti
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(0,10)
plt.xlim(0,6)
plt.title('Basic Resource Allocation')
plt.legend()
plt.grid(True)
plt.show()
```

### Solution And Graph

```
Optimal values:
x = 2.0
y = 2.0
Optimal objective function value (z) = 18.0
```



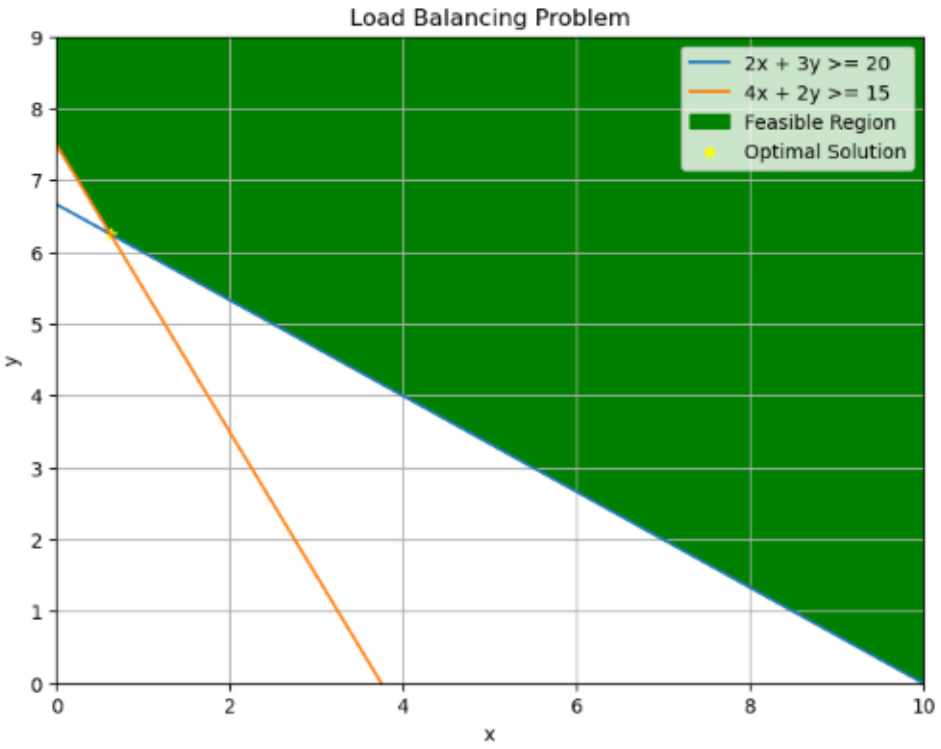
### 1.2 Number Two: Load Balancing

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import linprog

# Define the objective function coefficients
c = [5, 4]

# Coefficients of the inequality constraints
A = np.array([
[-2, -3],  # Server 1 constraint
[-4, -2],  # Server 2 constraint
])

# Right-hand side values of the inequality constraints
b = [-20, -15]

# Solve linear programming problem
result = linprog(c, A_ub=A, b_ub=b)

# Display the results
print("Optimal values:")
print("x =", result.x[0])
print("y =", result.x[1])
print("Optimal objective function value (z) =", result.fun)

# Plotting the feasible region
x_values = np.linspace(0, 10, 100)
y1_values = (20 - 2*x_values) / 3  # Server 1 constraint
y2_values = (15 - 4*x_values) / 2  # Server 2 constraint

plt.figure(figsize=(8, 6))
plt.plot(x_values, y1_values, label='2x + 3y >= 20')
plt.plot(x_values, y2_values, label='4x + 2y >= 15')


x = [0.625, 0, 0, 10, 10]
y = [6.25, 7.5, 10, 10, 0]
plt.fill(x,y, color="green", label="Feasible Region")

# plt.fill_between(x_values, np.maximum(y1_values, y2_values), color='gray', alpha=0
plt.scatter(result.x[0], result.x[1], color='yellow', marker='*', label='Optimal Sol
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(0, 10)
plt.ylim(0, 9.0)
plt.title('Load Balancing Problem')
plt.legend()
plt.grid(True)
plt.show()
```

### Solution And Graph

```
Optimal values:
x = 0.6250000000000003
y = 6.25
Optimal objective function value (z) = 28.125
```



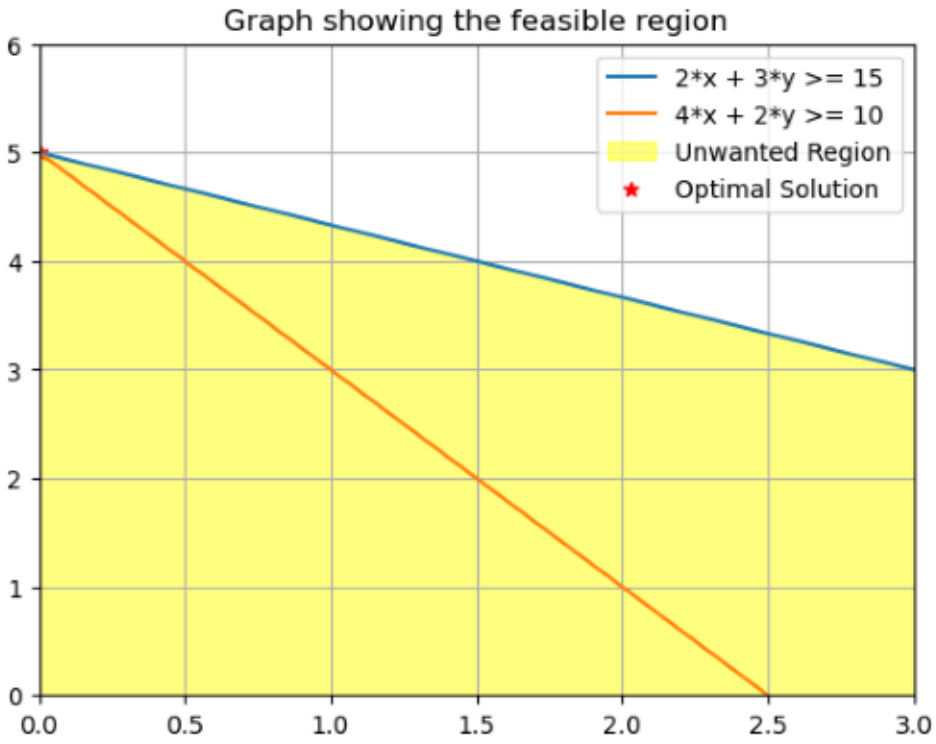
### 1.3 Number Three: Energy Efficient Resource Allocation

```
from pulp import *
import matplotlib.pyplot as plt
import numpy as np

model = LpProblem(name = "Energy_consumption", sense=LpMinimize)

x = LpVariable("x", 0)
y = LpVariable("y", 0)

model += 3*x + 2*y

model += 2*x + 3*y >= 15
model += 4*x + 2*y >= 10

model.solve()

optimal_x = x.varValue
optimal_y = y.varValue
optimal_value = model.objective.value()

print("Optimal Solution: ")
print("x: ", optimal_x)
print("y: ", optimal_y)
print("Objective value: ", optimal_value)


x_values = np.linspace(0, 10, 100)
y1_values = (15-2*x_values)/3
y2_values = (10-4*x_values)/2

plt.plot(x_values, y1_values, label=2*x + 3*y >= 15)
plt.plot(x_values, y2_values, label=4*x + 2*y >= 10)

# plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values<=y
plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values>=y2_


plt.title("Graph showing the feasible region")
plt.xlim(0,3)
plt.ylim(0,6)
plt.grid(True)
plt.scatter(optimal_x, optimal_y, color="red",marker='*', label="Optimal Solution")
```

```
plt.legend()
plt.show()
```

**Solution And Graph**

```
Optimal Solution:
x:  0.0
y:  5.0
Objective value:  10.0
```



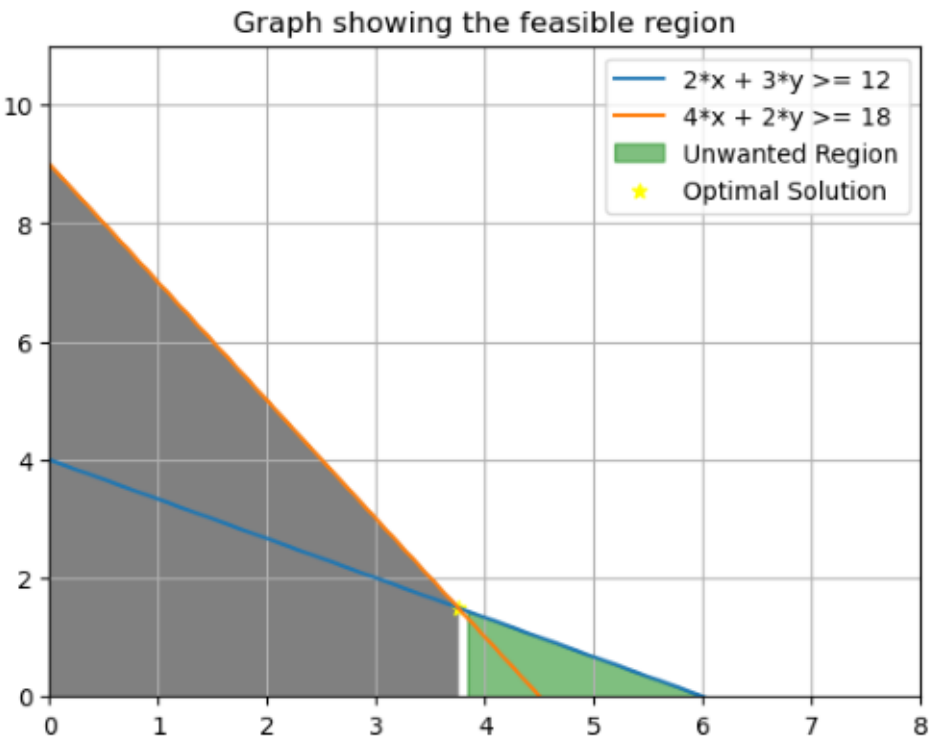
### 1.4 Number Four: Multi-Tenant Resource Sharing

```
from pulp import *
import matplotlib.pyplot as plt
import numpy as np

model = LpProblem(name = "Multi-Tenant-Resource-Sharing", sense=LpMinimize)

x = LpVariable("x", 0)
y = LpVariable("y", 0)

model += 5*x + 4*y

model += 2*x + 3*y >= 12
model += 4*x + 2*y >= 18

model.solve()

optimal_x = x.varValue
optimal_y = y.varValue
optimal_value = model.objective.value()

print("Optimal Solution: ")
print("x: ", optimal_x)
print("y: ", optimal_y)
print("Objective value: ", optimal_value)


x_values = np.linspace(0, 10, 100)
y1_values = (12-2*x_values)/3
y2_values = (18-4*x_values)/2

plt.plot(x_values, y1_values, label=2*x + 3*y >= 12)
plt.plot(x_values, y2_values, label=4*x + 2*y >= 18)

plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values<=y2_
plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values>=y2_


plt.title("Graph showing the feasible region")
plt.xlim(0,8)
plt.ylim(0,11)
plt.grid(True)
plt.scatter(optimal_x, optimal_y, color="yellow", marker='*', label="Optimal Solutio
```

```
plt.legend()
plt.show()
```

**Solution And Graph**

```
Optimal Solution:
x:  3.75
y:  1.5
Objective value:  24.75
```



# 2 Transportation Logistics Optimization

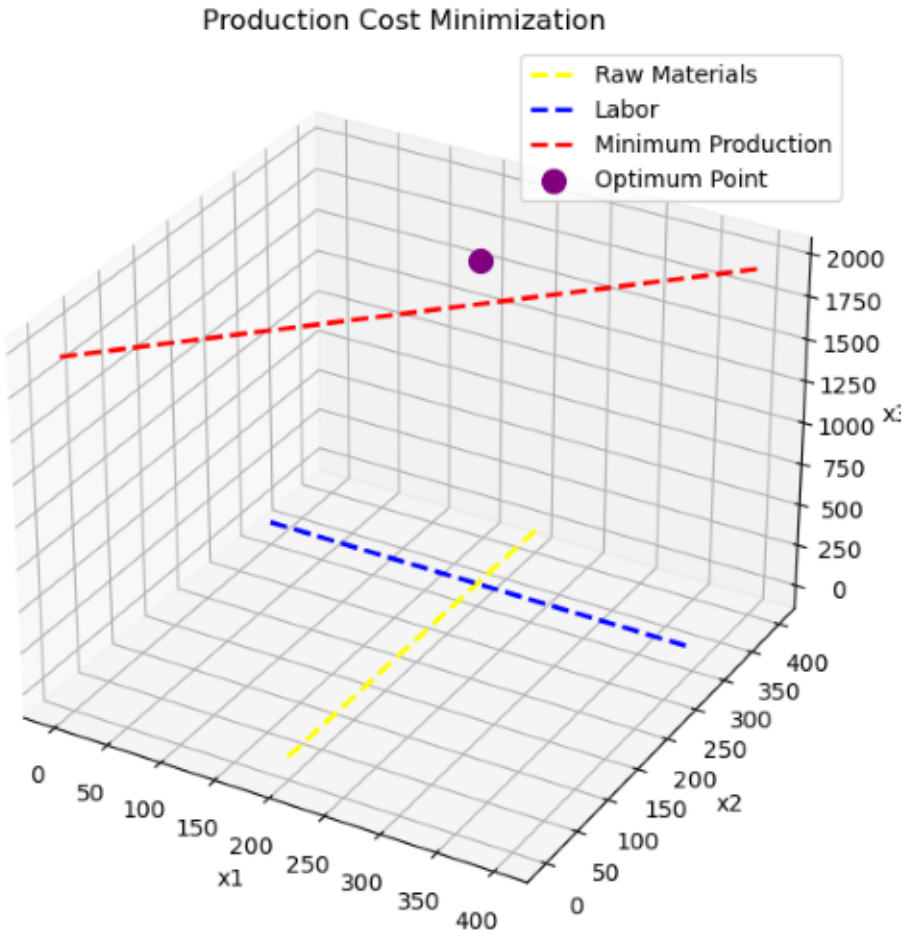
## 2.1 Number Five: Production Planning in Manufacturing

```
#libraries
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from pulp import LpVariable, LpMinimize, LpProblem

#the problem
problem = LpProblem(name="Production_minimizing", sense=LpMinimize)

#decision variables
x1 = LpVariable(name="x", lowBound=0)
x2 = LpVariable(name="y", lowBound=0)
x3 = LpVariable(name="z", lowBound=0)

# Define the objective function coefficients
problem += 5*x1 + 3*x2 + 4*x3, "objective"

# Coefficients of the inequality constraints
problem += 2*x1 + 3*x2 + x3 <= 1000, "raw_materials"
problem += 4*x1 + 2*x2 + 5*x3 <= 120, "labour"
problem += x1 >= 200
problem += x2 >= 300
problem += x3 >= 150


# Solve
problem.solve()

# Display the results
print("OPTIMAL SOLUTION:")
print(f"X: {x1.varValue}")
print(f"Y: {x2.varValue}")
print(f"Z: {x3.varValue}")
print(f"Minimum cost: {problem.objective.value()}")


# plotting the graph

# Create a meshgrid for x1, x2, and x3
x1_vals = np.linspace(0, 400, 50)
```

```
x2_vals = np.linspace(0, 400, 50)
x1_grid, x2_grid = np.meshgrid(x1_vals, x2_vals)

# Calculate the corresponding z-values (objective function)
z_vals = 5 * x1_grid + 3 * x2_grid + 4 * (1950 - x1_grid - x2_grid)

# Create the 3D plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Plot the feasible region (constraints)
ax.plot([200, 200], [0, 400], [0, 0], color='yellow', linestyle='--', linewidth=2, l
ax.plot([0, 400], [300, 300], [0, 0], color='blue', linestyle='--', linewidth=2, lab
ax.plot([0, 400], [0, 400], [1950, 1950], color='red', linestyle='--', linewidth=2,


optimum_x1 = 200
optimum_x2 = 300
optimum_z = 1950
ax.scatter(optimum_x1, optimum_x2, optimum_z, color='purple', s=100, label='Optimum


ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Production Cost Minimization')
ax.legend()
plt.show()
```

**Solution And Graph**

```
OPTIMAL SOLUTION:
X: 200.0
Y: 300.0
Z: 0.0
Minimum cost: 1900.0
```



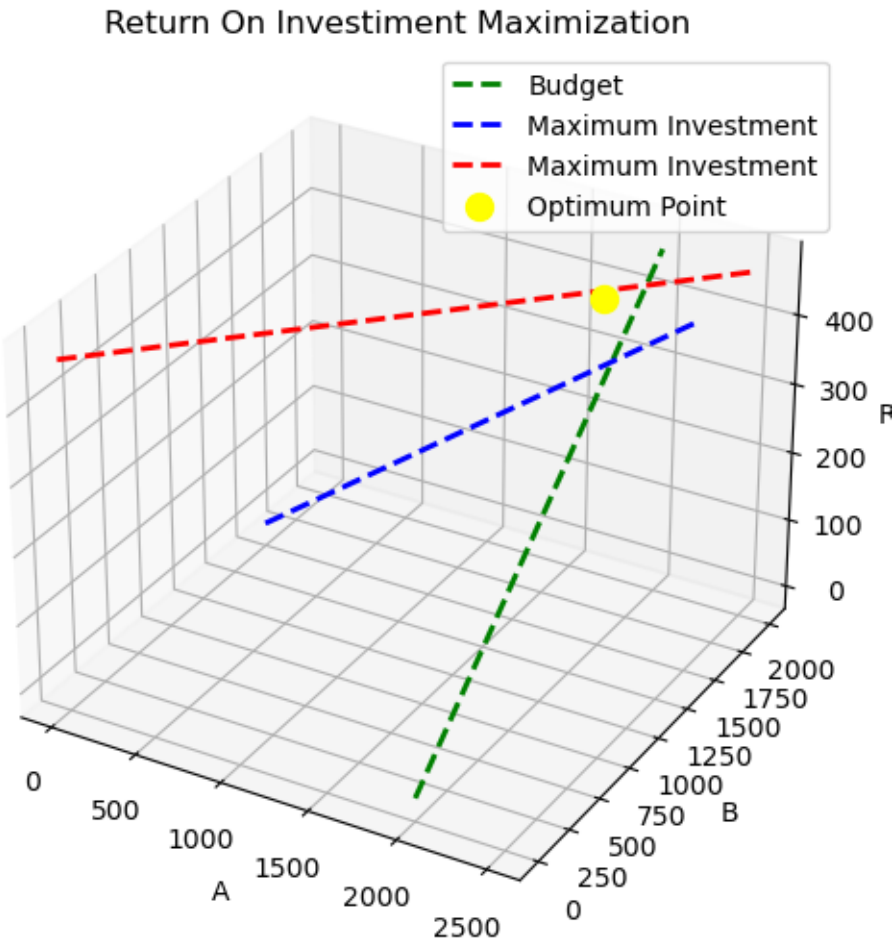
## 2.2 Number Six: Financial Portfolio Optimization

```
#libraries
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from pulp import LpVariable, LpMaximize, LpProblem

#the problem
problem = LpProblem(name="Financial_portfolio_optimization", sense=LpMaximize)

#decision variables
x1 = LpVariable(name="x", lowBound=0)
x2 = LpVariable(name="y", lowBound=0)
x3 = LpVariable(name="z", lowBound=0)

# Define the objective function coefficients
problem += 0.08*x1 + 0.1*x2 + 0.12*x3, "objective"

# Coefficients of the inequality constraints
problem += 2*x1 + 3*x2 + x3 <= 10000, "budget"
problem += x1 >= 2000
problem += x2 >= 1500
problem += x3 >= 1000


# Solve linear programming problem
problem.solve()

# Display the results
print("OPTIMAL SOLUTION:")
print(f"X: {x1.varValue}")
print(f"Y: {x2.varValue}")
print(f"Z: {x3.varValue}")
print(f"Minimum cost: {problem.objective.value()}")


# Create a meshgrid for A, B, and C
A_vals = np.linspace(0, 2500, 50)
B_vals = np.linspace(0, 2000, 50)
A_grid, B_grid = np.meshgrid(A_vals, B_vals)

# Calculate the corresponding z-values (ROI function)
C_vals = (10000 - 2 * A_grid - 3 * B_grid)  # Budget constraint
```

```
ROI_vals = 0.08 * A_grid + 0.1 * B_grid + 0.12 * C_vals

# Create the 3D plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the feasible region (constraints)
ax.plot([2000, 2000], [0, 2000], [0, 470], color='green', linestyle='--', linewidth=
ax.plot([0, 2500], [1500, 1500], [0, 470], color='blue', linestyle='--', linewidth=2
ax.plot([0, 2500], [0, 2000], [470, 470], color='red', linestyle='--', linewidth=2,

# Highlight the optimum point
optimum_A = 2000
optimum_B = 1500
optimum_ROI = 470
ax.scatter(optimum_A, optimum_B, optimum_ROI, color='yellow', s=100, label='Optimum

# Set labels and title
ax.set_xlabel('A')
ax.set_ylabel('B')
ax.set_zlabel('ROI')
ax.set_title('Return On Investiment Maximization')

# Add a legend
ax.legend()

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

**Solution And Graph**

```
OPTIMAL SOLUTION:
X: 2000.0
Y: 1500.0
Z: 1500.0
Minimum cost: 490.0
```



# 3 Project Resource Allocation Optimization

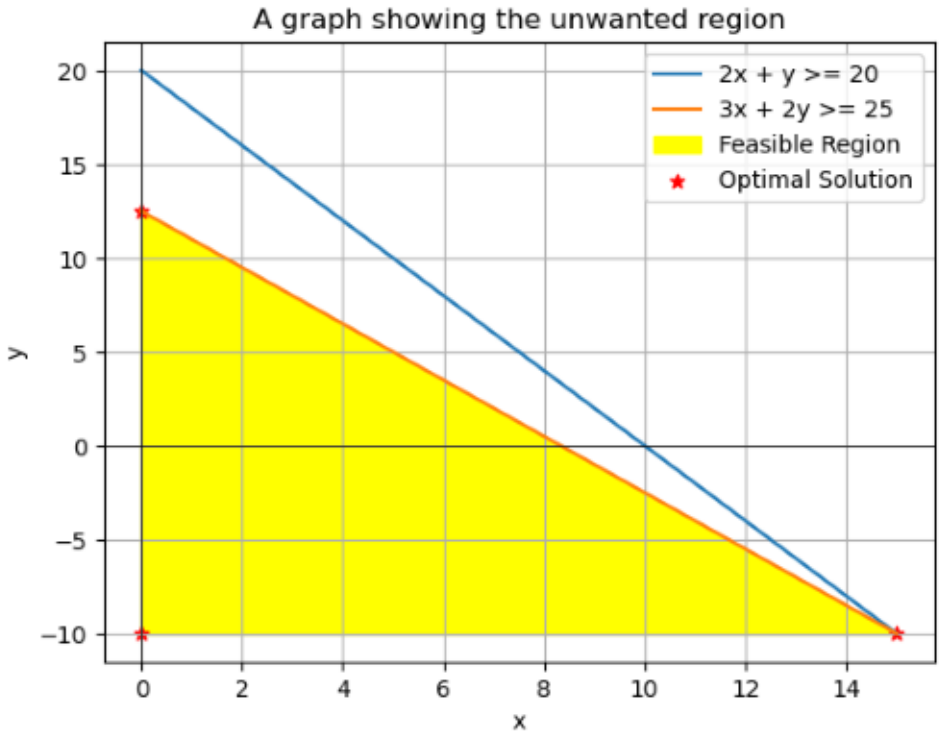
## 3.1 Number Seven: Diet Optimization

```
from pulp import*
import matplotlib.pyplot as plt
import numpy as np

#lpProblem
model = LpProblem(name="diet_optimization", sense=LpMinimize)

#LpVariables
x=LpVariable("x",0)
y=LpVariable("y",0)

#objective function
model+=3*x + 2*y

#constraints
model+= 2*x + y >= 20
model+=3*x + 2*y>= 25

#solving
model.solve()

#display results
x=x.varValue
y=y.varValue
optimal_value= model.objective.value()

print("status:",LpStatus[model.status])
print("x",x)
print("y",y)
print("optimal", optimal_value)



# Define the constraints
def constraint1(x):
return 20 - 2*x

def constraint2(x):
return (25 - 3*x) / 2
```

```
# Create x range for plotting
x_values = np.linspace(0, 15, 400)

# Plot constraints
plt.plot(x_values, constraint1(x_values), label='2x + y >= 20')
plt.plot(x_values, constraint2(x_values), label='3x + 2y >= 25')

# Shade unwanted region
x=[0,0,15]
y=[12.5,-10,-10]
plt.fill(x,y,color='gray')
plt.fill(x,y, color="yellow", label="Feasible Region")
plt.scatter(x, y, color='red', marker='*', label='Optimal Solution')


# Set labels and legend
plt.xlabel('x')
plt.ylabel('y')
plt.title('A graph showing the unwanted region')
plt.legend()

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

**Solution And Graph**

```
status: Optimal
x 10.0
y 0.0
optimal 30.0
```



### 3.2 Number Eight: Production Planning

```
from pulp import *
import matplotlib.pyplot as plt
import numpy as np

model = LpProblem(name = "Production-Planning", sense=LpMaximize)

x1 = LpVariable("x1", 0)
x2 = LpVariable("x2", 0)

model += 5*x1 + 3*x2

model += 2*x1 + 3*x2 <= 60
model += 4*x1 + 2*x2 <= 80

model.solve()

optimal_x = x1.varValue
optimal_y = x2.varValue
optimal_value = model.objective.value()

print("Optimal Solution: ")
print("x: ", optimal_x)
print("y: ", optimal_y)
print("Objective value: ", optimal_value)


#plotting the graph
x_values = np.linspace(0, 100, 1000)
y1_values = (60-2*x_values)/3
y2_values = (80-4*x_values)/2

plt.plot(x_values, y1_values, label=2*x1 + 3*x2 <= 60)
plt.plot(x_values, y2_values, label=4*x1 + 2*x2 <= 80)

plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values>=y2_
plt.fill_between(x_values, np.maximum(y1_values,y2_values), 0, where=(y1_values<=y2_


plt.title("Feasible region")
plt.xlim(0,38)
plt.ylim(0,50)
plt.grid(True)
```

```
plt.scatter(optimal_x, optimal_y, color="yellow", marker='*', label="Optimal Solutio
plt.legend()
plt.show()
```

**Solution And Graph**

```
Optimal Solution:
x:  15.0
y:  10.0
Objective value:  105.0
```