MOUNTAINS OF THE MOON UNIVERSITY FACULTY OF SCIENCE,TECHNOLOGY AND INNOVATION

NALUKANGA WINNIE [2023/U/MMU/BCS/00102]

10/FEBRUARY 2024

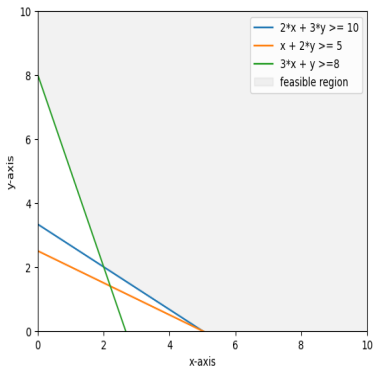
1 Solutions to Linear programming problems

1.1 basic resource allocation

```
    from pulp import LpProblem , LpMinimize ,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="resource_allocation",sense=LpMinimize)
#variables
x = LpVariable("x",0)
y = LpVariable("y",0)
#objective function
model+= 4*x +5*y
#constraints
model += 2*x + 3*y >= 10,"CPU constraint"
model += x + 2*y >= 5, "Memory constraint"
model += 3*x +y >= 8, "Storage constraint"
#solvinng
model.solve()
#results
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("z:",Optimal_value )
```

```
optimal solution:
x: 2.0
y: 2.0
z: 18.0
```

```
    #BASIC RESOURCE ALLOCATION(1)
import matplotlib.pyplot as plt
import numpy as np

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

# convert contraints 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
y4 = np.maximum.reduce([y1, y2, y3])
plt.fill_between(x , y4, 11, color="gray", label="feasible region", alpha=0.1)

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

# label and show graph

plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()

plt.show()
```



t

Figure 1: graph

1.2 load balancing

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Load_balancing",sense=LpMinimize)
#variables
x = LpVariable("x",0)
y = LpVariable("y",0)
#objective function
model+= 5*x +4*y
#constraints
model += 2*x + 3*y <= 20,"Server 1 constraint"
model += 4*x + 2*y <= 15, "Server 2 constraint"

#solvinng
model.solve()
#results
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("z:",Optimal_value )

optimal solution:
x: 0.0
y: 0.0
z: 0.0
```

graph codes

```
    #LOAD BALANCING(2)
import matplotlib.pyplot as plt
import numpy as np
# define the array
x= np.linspace(0,15,1000)
#convert the constraints to inqualities
y1=(20-2*x)/3
y2=(15-4*x)/2
#plot the constraints
plt.plot(x,y1 ,label="3*y+ 2*x<=20")
plt.plot(x,y2 ,label="2*y+ 4*x<=15")
#defining the feasible region
```

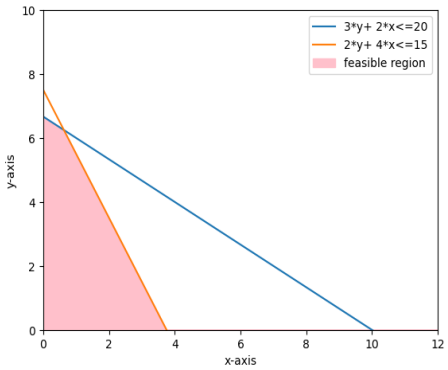


Figure 2: graph 2

```
y3=np.minimum.reduce([y1,y2])
plt.fill_between(x, y3,  color="pink",label=("feasible region"))
#limits
plt.xlim(0,12)
plt.ylim(0,10)
# label
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()

plt.show()
```

1.3 resource allocation

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="resource_allocation",sense=LpMinimize)
#variables
x = LpVariable("x",0)
y = LpVariable("y",0)
#objective function
model+= 3*x +2*y
#constraints
model += 2*x + 3*y >= 15,"CPU constraint"
model += 4*x + 2*y >= 10, "Memory constraint"

#solvinng
```

```
model.solve()
#results
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("z:",Optimal_value )

optimal solution:
x: 0.0
y: 5.0
z: 10.0


        #ENERGY EFFICIENCY(3)
import matplotlib.pyplot as plt
import numpy as np
# define the array
x= np.linspace(0,15,1000)
#convert the constraints to inqualities
y1=(15-2*x)/3
y2=(10-4*x)/2
#plot the constraints
plt.plot(x,y1 ,label="2*x + 3*y>=15")
plt.plot(x,y2 ,label="4*x+ 2*y>=10")
#defining the feasible region
y3=np.minimum.reduce([y1,y2])
plt.fill_between(x, y3, 6, color="pink",label=("feasible region"))
#limits
plt.xlim(0,12)
plt.ylim(0,10)
# label
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()

plt.show()
```

1.4 multi tenant resource sharing

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

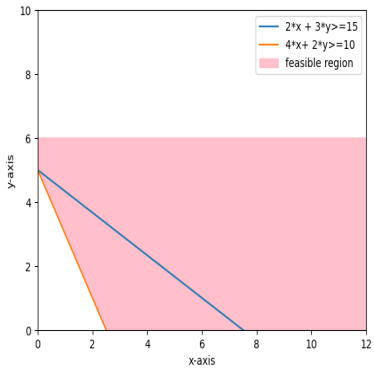


Figure 3: graph 3

```
#LpProblem
model = LpProblem(name="Multi_Tenant_Resource_Sharing",sense=LpMinimize)
#variables
x = LpVariable("x",0)
y = LpVariable("y",0)
#objective function
model+= 5*x +4*y
#constraints
model += 2*x + 3*y >= 12,"tenant 1"
model += 4*x + 2*y >= 18, "tenant 2"

#solvinng
model.solve()
#results
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("z:",Optimal_value )
optimal solution:
x1: 15.0
x2: 10.0
z: 105.0


#MULTI TENANT RESOURCE SHARING(4)
import matplotlib.pyplot as plt
import numpy as np
# define the array
```

```
x= np.linspace(0,15,1000)
#convert the constraints to inqualities
y1=(12-2*x)/3
y2=(18-4*x)/2
#plot the constraints
plt.plot(x,y1 ,label="2*x+ 3*y>=12")
plt.plot(x,y2 ,label="4*x+ 2*y>=18")
#defining the feasible region
y3=np.maximum.reduce([y1,y2])
plt.fill_between(x, y3, 10, color="pink",label=("feasible region"))
#limits
plt.xlim(0,12)
plt.ylim(0,10)
# label
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()

plt.show()
```

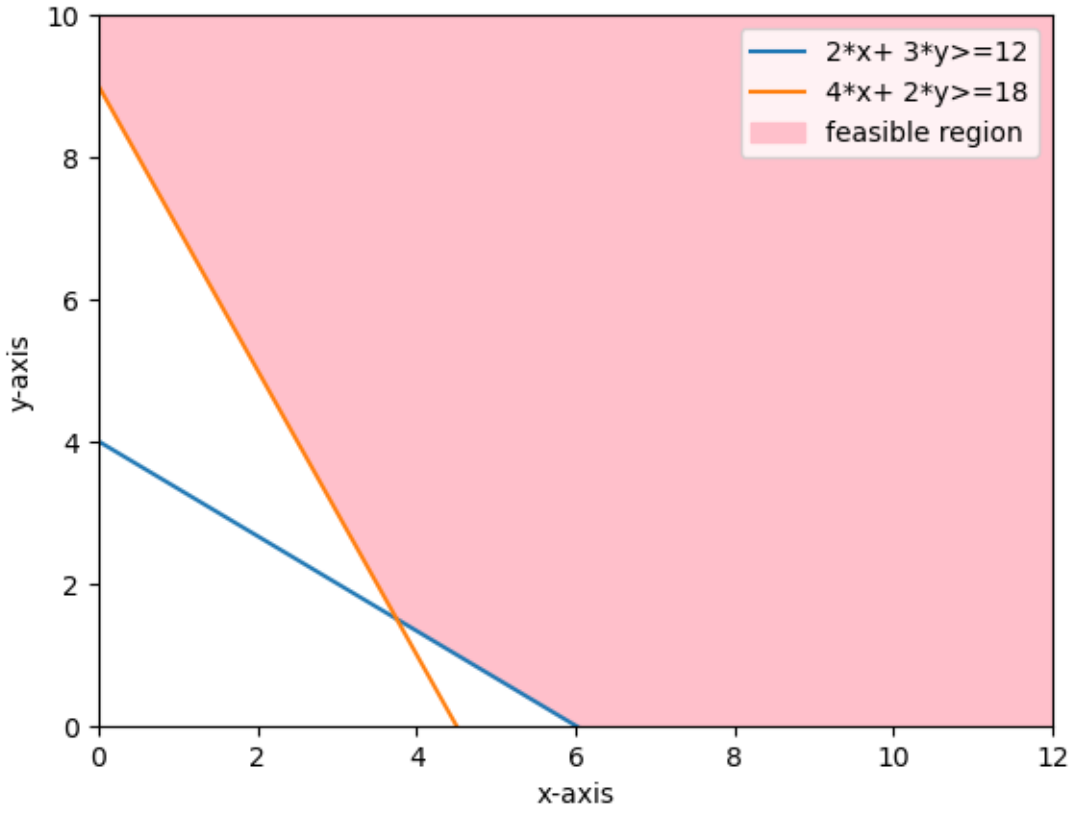


Figure 4: graph 4

1.5 production planning

```
    from pulp import LpProblem, LpMinimize, LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model=LpProblem("name=production_planning_manufacturing", sense=LpMinimize)
#LpVariables
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
x3=LpVariable("x3",0)
#objective function
model+=5*x1+ 3*x2 +4*x3
#constraints
model+=2*x1 +3*x2+ x3<=1000, "raw_materials"
model+=4*x1 +2*x2+ 5*x3<=120 ,"labour_hour"
model+=x1>=200,"minimum_x1"
model+=x2>=300,"minimum_x2"
model+=x3>=150,"minimum_x3"
#solving
model.solve()
#results
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_x3=x3.varValue
optimal_Value=model.objective.value()
print("optimum solution")
print("x1;",optimal_x1)
print("x2;",optimal_x2)
print("x3;",optimal_x3)
print("z;",optimal_Value)
optimum solution
x1; 200.0
x2; 300.0
x3; 0.0
z; 1900.0
```

1.6 financial portifolio

```
    from pulp import LpProblem, LpMaximize, LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model=LpProblem("name=production_planning_manufacturing", sense=LpMaximize)
```

```
#LpVariables
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
x3=LpVariable("x3",0)j
#objective function
model+=0.08*x1+ 0.1*x2 +0.12*x3
#constraints
model+=2*x1 +3*x2+ x3<=10000, "budget constraint"
model+=x1>=2000, "minimum_x1"
model+=x2>=1500, "minimum_x2"
model+=x3>=1000, "minimum_x3"
#solving
model.solve()
#results
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_x3=x3.varValue
optimal_Value=model.objective.value()
print("optimum solution")
print("x1;",optimal_x1)
print("x2;",optimal_x2)
print("x3;",optimal_x3)
print("z;",optimal_Value)
```

```
optimum solution
x1; 2000.0
x2; 1500.0
x3; 1500.0
z; 490.0
```

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

# Define the inequalities
def inequality1(x1, x2, x3):
    return 2*x1 + 3*x2 + x3

def inequality2(x1, x2, x3):
    return 4*x1 + 2*x2 + 5*x3

# Define the variable ranges
x1_range = np.linspace(200, 500, 100)  # Adjust the range based on your problem
x2_range = np.linspace(300, 500, 100)
```

```
x3_range = np.linspace(150, 300, 100)

# Generate a grid of points
X1, X2 = np.meshgrid(x1_range, x2_range)

# Evaluate the inequalities
Z1 = inequality1(X1, X2, 0)  # Set X3 to 0 for visualization
Z2 = inequality2(X1, X2, 0)

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

# Plot the surfaces
ax.plot_surface(X1, X2, Z1, alpha=0.5, rstride=100, cstride=100, color='r')
ax.plot_surface(X1, X2, Z2, alpha=0.5, rstride=100, cstride=100, color='g')

# Set labels
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('Z (X3 set to 0)')

# Set viewpoint
ax.view_init(elev=20, azim=-40)

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

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

# Define the inequalities
def inequality1(x1, x2, x3):
    return 2*x1 + 3*x2 + x3

def inequality2(x1, x2, x3):
    return 4*x1 + 2*x2 + 5*x3

# Define the variable ranges
x1_range = np.linspace(200, 500, 100)  # Adjust the range based on your problem
x2_range = np.linspace(300, 500, 100)
x3_range = np.linspace(150, 300, 100)
```

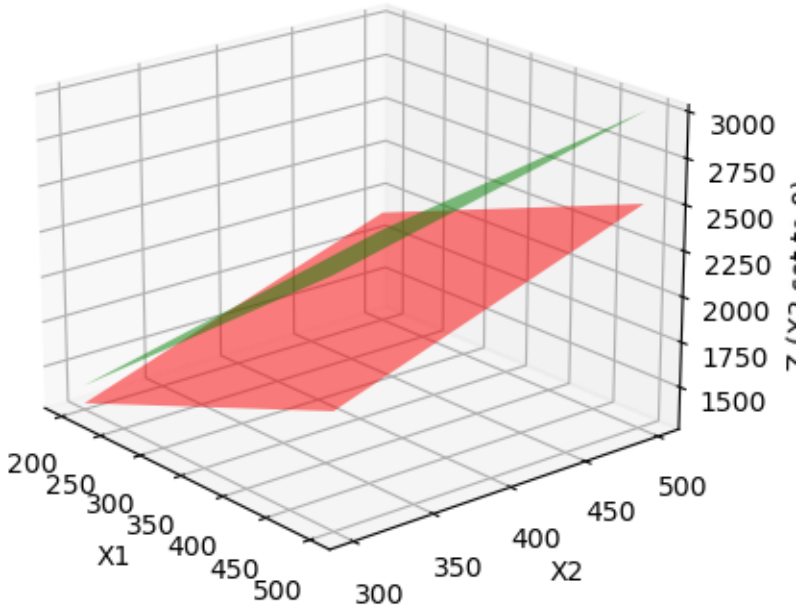


Figure 5: graph 5

```
# Generate a grid of points
X1, X2 = np.meshgrid(x1_range, x2_range)

# Evaluate the inequalities
Z1 = inequality1(X1, X2, 0)  # Set X3 to 0 for visualization
Z2 = inequality2(X1, X2, 0)

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

# Plot the surfaces
ax.plot_surface(X1, X2, Z1, alpha=0.5, rstride=100, cstride=100, color='r')
ax.plot_surface(X1, X2, Z2, alpha=0.5, rstride=100, cstride=100, color='g')

# Set labels
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('Z (X3 set to 0)')

# Set viewpoint
ax.view_init(elev=20, azim=-40)

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

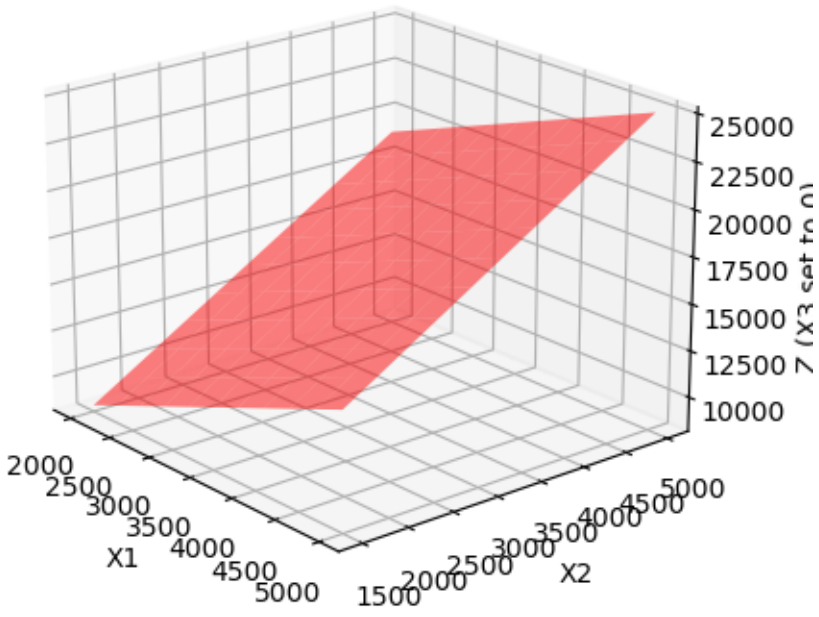


Figure 6: graph 6

1.7 diet optmization

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Diet_Optimization",sense=LpMinimize)
#variables
x1 = LpVariable("x1",0)
x2 = LpVariable("x2",0)
#objective function
model+= 3*x1 +2*x2
#constraints
model += 2*x1 + x2 >= 20,"proteins"
model += 3*x1 + 2*x2 >= 25, "vitamins"

#solvinng
model.solve()
#results
optimal_x1 = x1.varValue
optimal_x2 = x2.varValue
Optimal_value = model.objective.value()
print("optimal solution: ")
print("x1:", optimal_x1)
print("x2:", optimal_x2)
print("z:",Optimal_value )

optimal solution:
x1: 10.0
x2: 0.0
z: 30.0
```

```
    #DIET OPTIMIZATION(7)
import matplotlib.pyplot as plt
import numpy as np
# DEFINE THE X ARRAY
x= np.linspace(0,30,1000)
#convert the constraints to inqualities
x1=(20-2*x)
x2=(25-3*x)/2

#plot the constraints
plt.plot(x,x1 ,label= "2*x1 + x2>=20")
plt.plot(x,x2 ,label= "3*x1 + 2*x2>=25")

#defining the feasible region
x3=np.maximum.reduce([x1,x2])
```

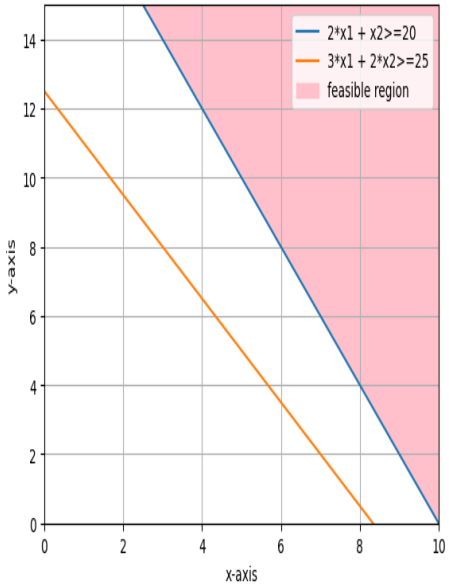


Figure 7: graph 7

```
plt.fill_between(x, x3, 15, color="pink",label=("feasible region"))
#limits
plt.xlim(0,10)
plt.ylim(0,15)
# label
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.grid()

plt.show()
```

1.8 profit

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="production_planning",sense=LpMaximize)
#variables
x1 = LpVariable("x1",0)
x2 = LpVariable("x2",0)
#objective function
model+= 5*x1 +3*x2
#constraints
model += 2*x1 + 3*x2 <= 60,"labour"
model += 4*x1 + 2*x2 <= 80, "raw_materials"

#solvinng
model.solve()
#results
optimal_x1 = x1.varValue
optimal_x2 = x2.varValue
Optimal_value = model.objective.value()
print("optimal solution: ")
print("x1:", optimal_x1)
print("x2:", optimal_x2)
print("z:",Optimal_value )
optimal solution:
x1: 15.0
x2: 10.0
z: 105.0
```

```
    # PRODUCTION PLANNING(8)
import matplotlib.pyplot as plt
import numpy as np
# DEFINE THE X ARRAY
x= np.linspace(0,50,1000)
#convert the constraints to inqualities
x1=(60-2*x)/3
x2=(80-4*x)/2

#plot the constraints
plt.plot(x,x1 ,label= "2*x1 +3*x2<=60")
plt.plot(x,x2 ,label= "4*x1 + 2*x2<=80")

#defining the feasible region
x3=np.minimum.reduce([x1,x2])
plt.fill_between(x, x3, color="blue",label=("feasible region"), alpha=.3)
```

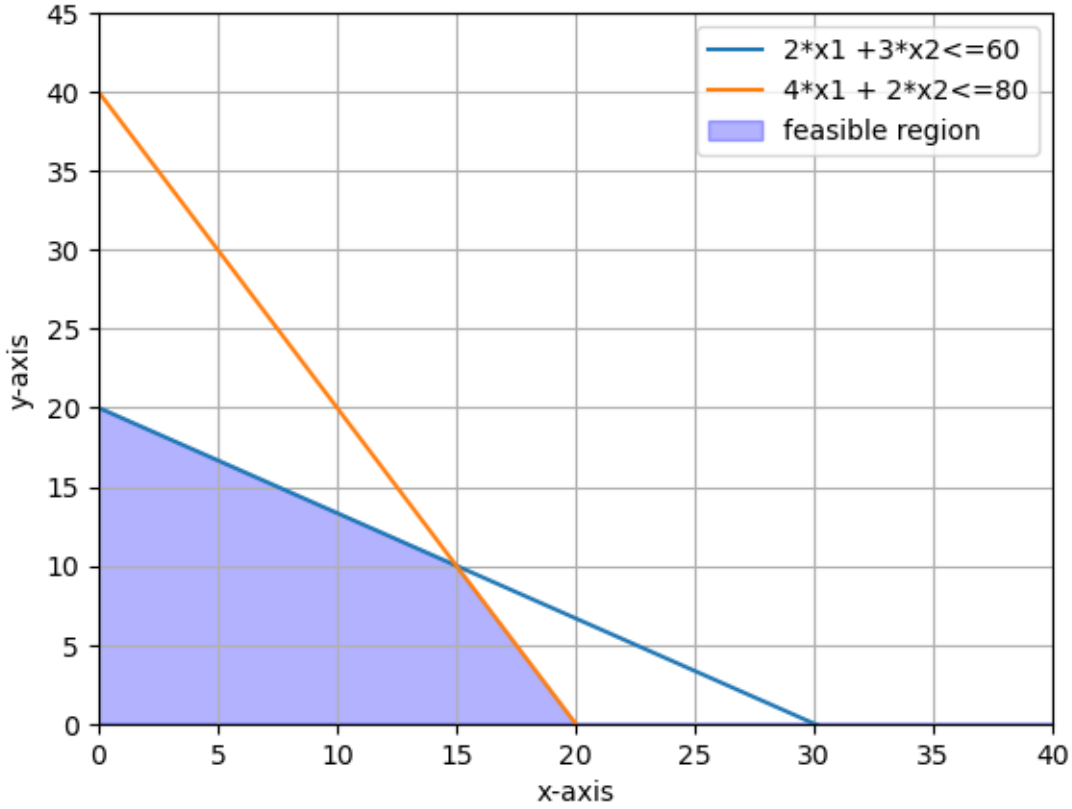


Figure 8: graph 8

```
#limits
plt.xlim(0,40)
plt.ylim(0,45)
# label
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.grid()

plt.show()
```