MOUNTAINS OF THE MOON UNIVERSITY**FACULTY OF SCIENCE, TECHNOLOGY AND INNOVATION**
DEPARTMENT OF COMPUTER SCIENCE
NAME:AKONYERA KATUSABE

- **REG.NO:2023/U/MMU/BCS/01573**

[

February 18, 2024

1 INTRODUCTION

No.1

Importing necessay Libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a LP minimization problem model = LpProblem(name="Production$_planning", sense = LpMaximize)$

Define Decision Variables x1 = LpVariable('x1', lowBound=0) Quantity of Product A x2 = LpVariable('x2', lowBound=0) Quantity of Product B

Define the objective function model += 4 * x1 + 2 * x2, "Objective"

Define inequalities as constraints model += 2 * x1 + 3 * x2 ¡= 60, "Labour$_{R}esource_{C}onstraint" model += 4 * x1 + 2 * x2 <= 80, "Raw_{M}aterial_{R}esource_{c}onstraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution for product planning:") print(f"Quantity of Product A (x1): x1.varValue") print(f"Quantity of Product B (x2): x2.varValue") print(f"Maximum Profit (Z): model.objective.value()")

Graph for No.1

import numpy as np import matplotlib.pyplot as plt

Define the inequalities def eq1(x): return (60 - 2*x) / 3

def eq2(x): return (80 - 4*x) / 2

Generate x values for plotting x$_{v}alues = np.linspace(0, 50, 400)$

Define the region boundaries

x$_{b}oundaries = np.linspace(0, 50, 10) y_{b}oundaries1 = eq1(x_{b}oundaries) y_{b}oundaries2 = eq2(x_{b}oundaries)$

Plotting the inequalities

plt.plot(x$_{v}alues, eq1(x_{v}alues), label = r'2x_1 + 3x_2 \leq 60$')

plt.plot(x$_{v}alues, eq2(x_{v}alues), label = r'4x_1 + 2x_2 \leq 80$')

Filling the unwanted regions

plt.fill$_{b}etween(x_{v}alues, 20, eq1(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 30, eq2(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 30, eq2(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 20, eq2(x_{v}alues), color =' gray', alpha = 0.5)$

Limiting the axes plt.xlim(0, 40) plt.ylim(0, 30)

Plot the points of intersection points = [(10, 20), (20, 0), (30, 0),(20, 0), (15, 10),(0, 20)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')

Labeling the axes plt.xlabel('x_1') plt.ylabel('x_2')

Adding grid plt.grid(True, linestyle='–')

Adding legend plt.legend() plt.title('production planning')

Show plot plt.show()

No.2

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Diet$_{O}ptimization", sense = LpMaximize$)

Define Decision Variables x1 = LpVariable(name="x1", lowBound=0) Quantity of product A x2 = LpVariable(name="x2", lowBound=0) Quantity of product B

Define the objective function model += 3 * x1 + 2 * x2, "Objective"

Define constraints model += 2 * x1 + x2 ¿= 20, "Proteins$_{N}utritional_{R}equirement_{C}onstraint" model+ = 3 * x1 + 2 * x2 >= 25, "Vitamins_{N}utritional_{r}equirenent_{C}onstraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution:") print(f'Quantity of Product A (x1): x1.varValue") print(f'Quantity of product B (x2): x2.varValue") print(f'Maximum Profit (Z): model.objective.value()")

Graph for No.2

import numpy as np import matplotlib.pyplot as plt

Define the inequalities def eq1(x): return (20 - 2*x) / 1

def eq2(x): return (25 - 3*x) / 2

Generate x values for plotting x$_{v}alues = np.linspace(0, 15, 400)$

Define the region boundaries

x$_{b}oundaries = np.linspace(0, 15, 10) y_{b}oundaries1 = eq1(x_{b}oundaries) y_{b}oundaries2 = eq2(x_{b}oundaries)$

Plotting the inequalities

plt.plot(x$_{v}alues, eq1(x_{v}alues), label = r'2\mathrm{x}_1 + x_2 \geq 20$')

plt.plot(x$_{v}alues, eq2(x_{v}alues), label = r'3\mathrm{x}_1 + 2x_2 \geq 25$')

Filling the unwanted regions

plt.fill$_{b}etween(x_{v}alues, 0, eq1(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 0, eq2(x_{v}alues), color =' gray', alpha = 0.5$)

Plot the points of intersection points = [(0, 12.5), (8.3, 0), (10, 0),(0, 20)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')

Limiting the axes plt.xlim(0, 20) plt.ylim(0, 25)

Labeling the axes plt.xlabel('x_1') plt.ylabel('x_2')

Adding grid plt.grid(True, linestyle='–')

Adding legend plt.legend() plt.title('Diet Optimization')

Show plot plt.show()

No.3

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Multi$_{T}enant_{R}esource_{P}lanning", sense = LpMaximize$)

Define Decision Variables x = LpVariable(name="x", lowBound=0) Quantity of product A y = LpVariable(name="y", lowBound=0) Quantity of product B

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

Define constraints model += 2 * x + 3 * y ¿= 12, "Tenant $1_C onstraint" model+ = 4 * x + 2 * y >= 18, "Tenant2_C onstraint"$
Solve the linear programming problem model.solve()
Display the results print("Optimal Solution:") print(f"Quantity of Product A (x): x.varValue") print(f"Quantity of product B (y): y.varValue")

Graph for No.3

import numpy as np import matplotlib.pyplot as plt
Define the inequalities def eq1(x): return (12 - 2*x) / 3
def eq2(x): return (18 - 4*x) / 2
Generate x values for plotting $x_v alues = np.linspace(0, 7, 400)$
Define the region boundaries
$x_b oundaries = np.linspace(0, 7, 10) y_b oundaries1 = eq1(x_b oundaries) y_b oundaries2 = eq2(x_b oundaries)$
Plotting the inequalities plt.plot($x_v alues, eq1(x_v alues), label = r'$2x+3y≥ 12')
plt.plot($x_v alues, eq2(x_v alues), label = r'$4x+2y≥ 18')
Filling the unwanted regions
plt.fill$_b etween(x_v alues, eq1(x_v alues), color =' gray', alpha = 0.5) plt.fill_b etween(x_v alues, eq2(x_v alues), color =' gray', alpha = 0.5)$
Limiting the axes plt.xlim(0, 7) plt.ylim(0, 10)
Plot the points of intersection points = [(3.75, 1.5), (6, 0), (0, 9)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')
Labeling the axes plt.xlabel('x') plt.ylabel('y')
Adding grid plt.grid(True, linestyle='–')
Adding legend plt.legend() plt.title('multi tenant resources sharing')
Show plot plt.show()

No.4

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Energy$_efficient_resource_allocation", sense = LpMaximize$)

Define Decision Variables x = LpVariable(name="x", lowBound=0) Quantity of product A y = LpVariable(name="y", lowBound=0) Quantity of product B

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

Define constraints model += 2 * x + 3 * y ¿= 15, "CPU allocation$_Constraint"model+ = 4*x+2*y >= 10, "Memory_Constraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution:") print(f'Quantity of Product A (x): x.varValue") print(f'Quantity of product B (y): y.varValue") print(f'Maximum Profit (Z): model.objective.value()")

Graph for No.4

import numpy as np import matplotlib.pyplot as plt

Define the inequalities def eq1(x): return (15 - 2*x) / 3

def eq2(x): return (10 - 4*x) / 2

Generate x values for plotting x$_values = np.linspace(0, 9, 400)$

Define the region boundaries

x$_boundaries = np.linspace(0, 7, 10)y_boundaries1 = eq1(x_boundaries)y_boundaries2 = eq2(x_boundaries)$

Plotting the inequalities plt.plot(x$_values, eq1(x_values), label = r'$2x+3y$\geq$ 15')

plt.plot(x$_values, eq2(x_values), label = r'$4x+2y$\geq$ 10')

Filling the unwanted regions

plt.fill$_between(x_values, 0, eq1(x_values), eq2(x_values), alpha = 0.5)plt.fill_between(x_values, 0, eq2(x_values), eq2(x_values), alpha = 0.5)$

Limiting the axes plt.xlim(0, 10) plt.ylim(0, 7)

Plot the points of intersection points = [(2.5, 0), (0, 5), (7.5, 0)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')

Labeling the axes plt.xlabel('x') plt.ylabel('y')

Adding grid plt.grid(True, linestyle='–')

Adding legend plt.legend() plt.title('Energy efficient resource allocation')

Show plot plt.show()

No.5

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable
Create a linear programming problem model = LpProblem(name="Loading$_{B}alance_{O}ptmization", sense = LpMaximize)$
Define Decision Variables x = LpVariable(name="x", lowBound=0) Quantity of product A y = LpVariable(name="y", lowBound=0) Quantity of product B
Define the objective function model += 5 * x + 4 * y, "Objective"
Define constraints model += 2 * x + 3 * y ¡= 20, "Server $1_{C}onstraint"model += 4 * x + 2 * y <= 15, "Server_{C}onstraint"$
Solve the linear programming problem model.solve()
Display the results print("Optimal Solution:") print(f"Quantity of Product A (x): x.varValue") print(f"Quantity of product B (y): y.varValue")
print(f"Maximum Profit (Z): model.objective.value()")

Graph for No.5

import numpy as np import matplotlib.pyplot as plt
Define the inequalities def eq1(x): return (20 - 2*x) / 3
def eq2(x): return (15 - 4*x) / 2
Generate x values for plotting x$_{v}alues = np.linspace(-1, 12, 400)$
Define the region boundaries
x$_{b}oundaries = np.linspace(-1, 12, 400) y_{b}oundaries1 = eq1(x_{b}oundaries) y_{b}oundaries2 = eq2(x_{b}oundaries)$
Plotting the inequalities plt.plot(x$_{v}alues, eq1(x_{v}alues), label = r'$2x+3y$\leq$ 20')
plt.plot(x$_{v}alues, eq2(x_{v}alues), label = r'$4x+2y$\leq$ 15')
Filling the unwanted regions
plt.fill$_{b}etween(x_{v}alues, 10, eq1(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 6.6, eq2(x_{v}alues), color =' gray', alpha = 0.5) plt.fill_{b}etween(x_{v}alues, 7.5, eq2(x_{v}alues), color =' gray', alpha = 0.5)$
Limiting the axes plt.xlim(0, 12) plt.ylim(0, 10)
Plot the points of intersection points = [(10, 0), (0, 6.6), (0, 7.5), (3.75, 0)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')
Labeling the axes plt.xlabel('x') plt.ylabel('y')
Adding grid plt.grid(True, linestyle='–')
Adding legend plt.legend() plt.title('loading balance optimization')
Show plot plt.show()

No.6

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Basic$_{r}$*esource$_{a}$llocation*", *sense* = *LpMaximize*)

Define Decision Variables x = LpVariable(name="x", lowBound=0) Quantity of product A y = LpVariable(name="y", lowBound=0) Quantity of product B

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

Define constraints model += 2 * x + 3 * y ¿= 10, "CPU$_{C}onstraint"model+ = x + 2 * y >= 5, "Memory_{C}onstraint"model+ = 3 * x + y >= 8, "Storage_{C}onstraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution:") print(f'Quantity of Product A (x): x.varValue") print(f'Quantity of product B (y): y.varValue") print(f'Maximum Profit (Z): model.objective.value()")

Graph for No.6

import numpy as np import matplotlib.pyplot as plt

Define the inequalities def eq1(x): return (10 - 2*x) / 3

def eq2(x): return (5 - x) / 2

def eq3(x): return 8 - 3*x

Generate x values for plotting x$_{v}alues = np.linspace(0, 5, 400)$

Define the region boundaries x$_{b}oundaries = np.linspace(0, 5, 10)y_{b}oundaries1 = eq1(x_{b}oundaries)y_{b}oundaries2 = eq2(x_{b}oundaries)y_{b}oundaries3 = eq3(x_{b}oundaries)$

Plotting the inequalities plt.plot(x$_{v}alues, eq1(x_{v}alues), label = r'$2x+3y$\geq$ 10')

plt.plot(x$_{v}alues, eq2(x_{v}alues), label = r'$x+2y$\geq$ 5')

plt.plot(x$_{v}alues, eq3(x_{v}alues), label = r'$3x+y$\geq$ 8')

Filling the unwanted regions

plt.fill$_{b}etween(x_{v}alues, eq1(x_{v}alues), color =' gray', alpha = 0.5)plt.fill_{b}etween(x_{v}alues, eq2(x_{v}alues), color =' gray', alpha = 0.5)plt.fill_{b}etween(x_{v}alues, eq3(x_{v}alues), color =' gray', alpha = 0.5)$

Limiting the axes plt.xlim(0, 6) plt.ylim(0, 10)

Plot the points of intersection points = [(5, 0), (0, 8), (2.2, 1.4),(2, 2)] for point in points: plt.plot(point[0], point[1], 'ro') plt.text(point[0], point[1], f'(point[0], point[1])', verticalalignment='bottom')

Labeling the axes plt.xlabel('x') plt.ylabel('y')

Adding grid plt.grid(True, linestyle='–')

Adding legend plt.legend() plt.title('Basic Resource maximization')

Show plot plt.show()

No.7

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Product$_{P}lanning_{i}n_{M}anufracturing$", $sense = LpMaximize$)

Define Decision Variables x1 = LpVariable(name="x1", lowBound=0) Quantity of product A x2 = LpVariable(name="x2", lowBound=0) Quantity of product B x3 = LpVariable(name="x3", lowBound=0) Quantity of product C

Define the objective function model += 5 * x1 + 3 * x2 + 4 * x3, "Objective"

Define constraints model += 2 * x1 + 3 * x2 + x3 ¡= 1000, "Raw$_{m}aterial_{C}onstraint$"$model+ = 4*x1+2*x2+5*x3 <= 120,$"$Labour_{h}ours_{C}onstraint$"$model+ = x1 >= 200,$"$Demand1_{c}onstraint$"$model+ = x2 >= 300,$"$Demand2_{c}onstraint$"$model+ = x1 >= 150,$"$Demand3_{c}onstraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution:") print(f'Quantity of Product A (x): x1.varValue") print(f'Quantity of product B (y): x2.varValue") print(f'Quantity of product C (y): x2.varValue") print(f"Maximum Profit (Z): model.objective.value()")

Graph for No.7

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model = LpProblem(name="Product$_{P}lanning_{i}n_{M}anufracturing$", $sense = LpMaximize$)

Define Decision Variables x1 = LpVariable(name="x1", lowBound=0) Quantity of product A x2 = LpVariable(name="x2", lowBound=0) Quantity of product B x3 = LpVariable(name="x3", lowBound=0) Quantity of product C

Define the objective function model += 5 * x1 + 3 * x2 + 4 * x3, "Objective"

Define constraints model += 2 * x1 + 3 * x2 + x3 ¡= 1000, "Raw$_{m}aterial_{C}onstraint$"$model+ = 4*x1+2*x2+5*x3 <= 120,$"$Labour_{h}ours_{C}onstraint$"$model+ = x1 >= 200,$"$Demand1_{c}onstraint$"$model+ = x2 >= 300,$"$Demand2_{c}onstraint$"$model+ = x1 >= 150,$"$Demand3_{c}onstraint$"

Solve the linear programming problem model.solve()

Display the results print("Optimal Solution:") print(f'Quantity of Product A (x): x1.varValue") print(f'Quantity of product B (y): x2.varValue") print(f'Quantity of product C (y): x2.varValue") print(f"Maximum Profit (Z): model.objective.value()")

No.8

Importing necessary libraries from pulp import LpProblem, LpMaximize, LpVariable

Create a linear programming problem model =
LpProblem(name="Financial$_Portifolio_Optimization", sense = LpMaximize$)
Define Decision Variables x1 = LpVariable(name="x1", lowBound=0)
Quantity of product A x2 = LpVariable(name="x2", lowBound=0) Quantity
of product B x3 = LpVariable(name="x3", lowBound=0) Quantity of
product C
Define the objective function model += 0.08 * x1 + 0.1 * x2 + 0.12 * x3,
"Objective"
Define constraints model += 2 * x1 + 3 * x2 + x3 ¡= 10000,
"Budget$_Constraint"model+=x1>=$
$2000,"Minimum_investment_constraint1_constraint"model+=x2>=$
$1500,"Minimum_investment_constraint2_constraint"model+=x3>=$
$1000,"Minimum_investment_constraint3_constraint"$
Solve the linear programming problem model.solve()
Display the results print("Optimal Solution:") print(f"Quantity of Product A
(x): x1.varValue") print(f"Quantity of product B (y): x2.varValue")
print(f"Quantity of product C (y): x2.varValue") print(f"Maximum Profit (Z):
model.objective.value()")

Graph for No.8

import numpy as np import matplotlib.pyplot as plt from
mpl$_toolkits.mplot3dimportAxes3D$
Define the inequalities def inequality1(x1, x2, x3): return 2*x1 + 3*x2 + x3
¡= 10000
def inequality2(x1, x2, x3): return x1 ¿= 2000
def inequality3(x1, x2, x3): return x2 ¿= 1500
def inequality4(x1, x2, x3): return x3 ¿=0000
Define the ranges for x1, x2, and x3 x1 = np.linspace(0, 5000, 100) x2 =
np.linspace(0, 2000, 100) x3 = np.linspace(0, 15000, 100)
X1, X2, X3 = np.meshgrid(x1, x2, x3)
Create boolean mask for unwanted regions mask =
np.ones$_like(X1, dtype = bool)$
inequalities = [inequality1, inequality2, inequality3, inequality4] for inequality
in inequalities: mask = inequality(X1, X2, X3)
Plot the unwanted regions fig = plt.figure(figsize=(10, 8)) ax =
fig.add$_subplot(111, projection='3d')ax.voxels(mask, edgecolor='k')$
Set labels and title
ax.set$_xlabel('x1')ax.set_ylabel('x2')ax.set_zlabel('x3')ax.set_title('GraphofFinancialPortifolioOptimization')$
Show plot plt.show()