Homework 3

Thursday, February 8, 2024 11:35 AM

STOR415: INTRODUCTION TO OPTIMIZATION
DEPARTMENT OF STATISTICS AND OPERATIONS RESEARCH
SPRING 2024

INSTRUCTOR: MICHAEL O'NEILL

HOMEWORK 3: MULTIPERIOD AND TRANSPORTATION PROBLEMS

- For this homework, you will need to submit both handwritten work and the output and code from Jupyter notebooks. Gradescope only allows the submission of a single PDF, however, you can combine multiple PDFs into one using free tools online or Adobe Acrobat, which all UNC students should have access to. When submitting this assignment, please assign pages to specific questions to make it easier to grade.
- For each coding problem, create an ipynb with exactly the same name as is required in the problem. In the Julia code, declare variables with the name given in the problem. Then, after solving the problem, in the last cell of your notebook print (or use @show) all the values of all of the variables in your optimization problem as well as the value of the objective function.
- Please comment add comments to your code describing the variables, constraints and objective function of your model.
- Ensure that your notebook runs properly before submitting it. In the main bar, perform **Clear Outputs of All Cells** then **Run All** to ensure that there are no errors.
- To generate a PDF of your notebook:
  - In the main bar, click **Export** (may be hidden behind a 3 dots dropdown menu)
  - Choose Export as PDF (may require additional extensions).
  - If Export as PDF fails or does not give proper output, export the file as HTML, open this HTML file in a web browser and save the HTML file as a PDF.
  - If you have issues with this, first open the folder containing your notebooks in VSCode (instead of just the notebook itself), then try exporting to PDF. This often fixes issues that may otherwise occur.
  - If you cannot get exporting to work in VSCode, you may use a Jupyter Notebook to PDF converter online.

STOR415: Introduction to Optimization - Fall 2022 Student's name:_____

Question 1. (**30 points**):

The CRUD chemical plant produces as part of its production process a noxious compound called chemical X. Chemical X is highly toxic and needs to be disposed of properly. Fortunately, CRUD is linked by a pipe system to the FRESHAIR recycling plant that can safely reprocess chemical X. On any give day, the CRUD plant will produce the following amount of Chemical X (the plant operates between 9am and 3pm only):

Time	9-10 AM	10-11 AM	11 AM-12 PM	12-1 PM	1-2 PM	2-3 PM
Chemical X (in liters)	300	240	600	200	300	600

Because of environmental regulation, at no point in time is the CRUD plant allowed to keep more than 1000 litres on site and no chemical X is allowed to be kept overnight. At the top of every hour, at most 650 liters of chemical X can be sent to the FRESHAIR recycling plant. The cost of recycling chemical X is different for every hour:

Time	10 AM	11 AM	12 PM	1 PM	2 PM	3 PM
Chemical X (in liters)	30	40	35	45	38	50

You need to decide how much chemical X to send from the CRUD plant to the FRESHAIR recycling plant at the top of each hour, so that you can minimize the total recycling cost but also meet all the environmental constraints.

a) Formulate this problem as a linear program.

b) Implement and solve this linear program in Julia. To check correctness, the optimal objective value is 88,050. Make sure to print out the optimal objective value and the value of all decision variables.

Question 2. (**35 points**): Paul Hollywood bakes two types of cakes: cheesecakes and black forest cakes. During any week, he can bake at most 65 cakes in total. The costs per cake and the demands for cakes, which must be met in time, are given in the following table.

	Week 1		Week 2		Week 3	
Item	Demand	Cost/cake($)	Demand	Cost/cake($)	Demand	Cost/cake($)
Cheesecake	40	3.00	30	3.40	20	3.80
Black Forest	20	2.50	30	2.80	10	3.40

We assume that cakes baked during a month can be used to meet demand for this week. At the end of

each week (after all cakes have been baked and the current week's demand has been satisfied), a holding cost of 50 cents per cheesecake and 40 cents per black forest cake is incurred for cakes left in inventory. Those cakes can be used to satisfy future demand.

a) Formulate an LP to minimize the total cost of meeting the next three weeks' demands.

b) Implement your linear program and solve it using JuMP. To check the correctness of your solution: the optimal value is \$464.5. Make sure to print out the optimal objective value and the value of all decision variables.

Question 3. (35 points): A shipping company supplies goods to three customers, who require 40, 50 and 40 units respectively. The company has three warehouses, each of which has 30 units available. The costs of shipping 1 unit from each warehouse to each customer are shown in the table below.

From/To	Customer 1	Customer 2	Customer 3
Warehouse 1	\$15	\$35	\$25
Warehouse 2	\$10	\$50	\$40
Warehouse 3	\$20	\$40	\$30

There is a penalty for unmet demand: With customer 1, a penalty cost of \$70 per unit is incurred; with customer 2, \$75 per unit; and with customer 3, \$65 per unit. The company's goal is to minimize the total cost.

a) Formulate this problem as a balanced transportation problem by adding an appropriate dummy node to the transportation network.

b) Implement and solve the linear program from part a in JuMP. Display the values of all decision variables. (The optimal value is \$4,950.)

The end

1. a) X_i : amount of Chemical X sent from the CRUD plant to the FRESHAIR recycling plant at time $i \in [1,6]$

p_i : production of Chemical X at hour $i \in [1,6]$

S_i : Storage at the end of hour $i \in [1,6]$

$$\min z = \sum_{i=1}^{6} C_i X_i$$

$$\text{S.t. } S_{i-1} + p_i - X_i = S_i \quad \forall i \in \{[1,6]\}$$

$$S_i \leq 1{,}000 \quad \forall i \in \{[1,6]\}$$

$$X_i \leq 650 \quad \forall i \in \{[1,6]\}$$

$$S_0, S_6 = \emptyset$$

$$X, S \geq \emptyset \quad \forall i \in \{[1,6]\}$$

$x_i, s_i \geq 0 \quad \forall i \in \{[1,6]\}$

$$P = \begin{bmatrix} 300 \\ 240 \\ 600 \\ 200 \\ 300 \\ 600 \end{bmatrix} \qquad C = \begin{bmatrix} 30 \\ 40 \\ 35 \\ 45 \\ 38 \\ 50 \end{bmatrix}$$

2. a) X_{Ci} : amount of cheese cakes bake d in week $i \in [1,3]$

X_{bi} : amount of black forest cakes baked in week $i \in [1,3]$

h_{Ci} : amount of cheese cakes held in inventory at the end of week $i \in [1,3]$

h_{bi} : amount of black forest cakes held in inventory at the end of week $i \in [1,3]$

$$C_C = \begin{bmatrix} \$3.00 \\ \$3.40 \\ \$3.80 \end{bmatrix} \qquad C_b = \begin{bmatrix} \$2.50 \\ \$2.80 \\ \$3.40 \end{bmatrix}$$

$$D_c = \begin{bmatrix} 40 \\ 30 \\ 20 \end{bmatrix} \qquad D_b = \begin{bmatrix} 20 \\ 30 \\ 10 \end{bmatrix}$$

$$\min z = \sum_{i}^{3} \left(C_{Ci} X_{Ci} + C_{bi} X_{bi} + 0.5 h_{Ci} + 0.4 h_{bi} \right)$$

$i=1$

$$X_{ci} + h_{c(i-1)} \geq D_{ci}$$

S.t. $$X_{bi} + h_{b(i-1)} \geq D_{bi}$$

$$X_{ci} + h_{c(i-1)} - D_{ci} = h_{ci}$$

$$X_{bi} + h_{b(i-1)} - D_{bi} = h_{bi}$$

$$h_{c\emptyset}, h_{b\emptyset} = \emptyset$$

$$X_{ci} + X_{bi} \leq 65$$

$$X_{ci}, X_{bi}, h_{ci}, h_{bi} \geq \emptyset$$

3. a) X_{ij} : units shipped from warehouse i to customer j

$i \in [1, 3]$, $j \in [1, 3]$

d_j : dummy warehouse representing unmet demand for customer j

$$D = \begin{bmatrix} 40 \\ 50 \\ 40 \end{bmatrix} \quad C = \begin{bmatrix} \$15 & \$35 & \$25 \\ \$10 & \$50 & \$40 \\ \$20 & \$40 & \$30 \end{bmatrix}$$

$$P = \begin{bmatrix} 70 \\ 75 \end{bmatrix}$$

$$\lfloor 65 \rfloor$$

$$\min z = \sum_{i=1}^{3} \sum_{j=1}^{3} C_{ij} X_{ij} + \sum_{j=1}^{3} P_j d_j$$

$$\sum_{j=1}^{3} X_{ij} \leq 30 \quad \forall i \in [1,3]$$

$$\sum_{i=1}^{3} X_{ij} + d_j = D_j \quad \forall j \in [1,3]$$

$$d_j, X_{ij} \geq \emptyset \quad \forall i,j \in [1,3]$$

```julia
using JuMP, HiGHS

println("HOMEWORK 3:\n")

###Question 1###
println("Question 1\n")
println("b.\n")

# Model
model = Model(HiGHS.Optimizer)

# Decision Variables
@variable(model, 0 <= x[1:6] <= 650)
@variable(model, 0 <= s[0:6] <= 1000)

# Constants
production = [300, 240, 600, 200, 300, 600]
costs = [30, 40, 35, 45, 38, 50]

# Objective
@objective(model, Min, sum(costs[i] * x[i] for i in 1:6))

# Constraints
for i in 1:6
    @constraint(model, s[i-1] + production[i] - x[i] == s[i])
end
@constraint(model, s[0] == 0)
@constraint(model, s[6] == 0)

# Solve
optimize!(model)

# Results
println("Objective value: ", objective_value(model))
for i in 1:6
    println("x[$i] = ", value(x[i]))
    println("s[$i] = ", value(s[i]))
end

###Question 2###
println("Question 2\n")
println("b.\n")

# Model
model = Model(HiGHS.Optimizer)

# Constants
# Cost vectors
C_c = [3.00, 3.40, 3.80]
C_b = [2.50, 2.80, 3.40]
```

```
# Demand vectors
D_c = [40, 30, 20]
D_b = [20, 30, 10]

# Holding costs
H_c = 0.5
H_b = 0.4

# Decision variables
@variable(model, x_c[1:3] >= 0) # Cheesecakes baked
@variable(model, x_b[1:3] >= 0) # Black Forest cakes baked
@variable(model, h_c[0:3] >= 0) # Cheesecakes held in inventory
@variable(model, h_b[0:3] >= 0) # Black Forest cakes held in inventory

# Initialize inventory at time 0 to 0
@constraint(model, h_c[0] == 0)
@constraint(model, h_b[0] == 0)

# Objective function to minimize cost
@objective(model, Min, sum(C_c[i] * x_c[i] + C_b[i] * x_b[i] + H_c *
h_c[i] + H_b * h_b[i] for i in 1:3))

# Constraints for weeks 1 through 3
for i in 1:3
    # Demand must be met each week
    @constraint(model, x_c[i] + h_c[i-1] >= D_c[i])
    @constraint(model, x_b[i] + h_b[i-1] >= D_b[i])

    # Inventory balance constraints
    @constraint(model, x_c[i] - D_c[i] + h_c[i-1] == h_c[i])
    @constraint(model, x_b[i] - D_b[i] + h_b[i-1] == h_b[i])

    # Baking constraints
    @constraint(model, x_c[i] + x_b[i] <= 65)
end

# Solve the model
optimize!(model)

# Output the results
println("Objective value: ", objective_value(model))
for i in 1:3
    println("Week ", i, ":")
    println(" Cheesecakes baked: ", value(x_c[i]))
    println(" Black Forest cakes baked: ", value(x_b[i]))
    println(" Cheesecakes held in inventory: ", value(h_c[i]))
    println(" Black Forest cakes held in inventory: ", value(h_b[i]))
end
```

```julia
###Question 3###
println("Question 3\n")
println("b.\n")

# Constants
# Demand vector
D = [40, 50, 40]

# Cost matrix
C = [
    15 35 25
    10 50 40
    20 40 30
]
# Penalty costs vector
P = [70, 75, 65]

# Model
model = Model(HiGHS.Optimizer)

# Decision variables
@variable(model, x[1:3, 1:3] >= 0)  # Units shipped from warehouses to customers
@variable(model, d[1:3] >= 0)        # Dummy variables for unmet demand

# Objective function
@objective(model, Min, sum(C[i,j] * x[i,j] for i in 1:3, j in 1:3) + sum(P[j] * d[j] for j in 1:3))

# Constraints
for i in 1:3
    # Supply constraints for warehouses
    @constraint(model, sum(x[i,j] for j in 1:3) <= 30)
end

for j in 1:3
    # Demand constraints for customers
    @constraint(model, sum(x[i,j] for i in 1:3) + d[j] == D[j])
end

# Solve the model
optimize!(model)

# Output the results
println("Objective value: ", objective_value(model))
for i in 1:3
    for j in 1:3
        println("x[$i,$j] = ", value(x[i,j]))  # Units shipped from warehouse i to customer j
```

```
    end
    println("d[$i] = ", value(d[i]))  # Unmet demand for customer i
end
```

HOMEWORK 3:

Question 1

b.

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
6 rows, 11 cols, 16 nonzeros
4 rows, 9 cols, 12 nonzeros
Presolve : Reductions: rows 4(-4); columns 9(-4); elements 12(-8)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Ph1: 0(0) 0s
          7     8.8050000000e+04 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 7
Objective value     :  8.8050000000e+04
HiGHS run time      :          0.00
Objective value: 88050.0
x[1] = 300.0
s[1] = 0.0
x[2] = 40.0
s[2] = 200.0
x[3] = 650.0
s[3] = 150.0
x[4] = 0.0
s[4] = 350.0
x[5] = 650.0
s[5] = 0.0
x[6] = 600.0
s[6] = -0.0
Question 2

b.

Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
11 rows, 10 cols, 26 nonzeros
11 rows, 10 cols, 26 nonzeros
Presolve : Reductions: rows 11(-6); columns 10(-4); elements 26(-12)
Solving the presolved LP
Using EKK dual simplex solver - serial
```

```
  Iteration        Objective     Infeasibilities num(sum)
          0     1.7000039604e+02 Pr: 8(180) 0s
          5     4.6450000000e+02 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 5
Objective value     :  4.6450000000e+02
HiGHS run time      :          0.00
Objective value:

464.5
Week 1:
 Cheesecakes baked: 45.0
 Black Forest cakes baked: 20.0
 Cheesecakes held in inventory: 5.0
 Black Forest cakes held in inventory: 0.0
Week 2:
 Cheesecakes baked: 25.0
 Black Forest cakes baked: 40.0
 Cheesecakes held in inventory: 0.0
 Black Forest cakes held in inventory: 10.0
Week 3:
 Cheesecakes baked: 20.0
 Black Forest cakes baked: 0.0
 Cheesecakes held in inventory: 0.0
 Black Forest cakes held in inventory: 0.0
Question 3

b.


Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
6 rows, 12 cols, 21 nonzeros
6 rows, 12 cols, 21 nonzeros
Presolve : Reductions: rows 6(-0); columns 12(-0); elements 21(-0) -
Not reduced
Problem not reduced by presolve: solving the LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Pr: 3(130) 0s
          9     4.9500000000e+03 Pr: 0(0) 0s
Model   status      : Optimal
Simplex   iterations: 9
Objective value     :  4.9500000000e+03
HiGHS run time      :          0.00
Objective value: 4950.0
x[1,1] = 0.0
x[1,2] = 0.0
x[1,3] = 30.0
```

```
d[1] = 0.0
x[2,1] = 30.0
x[2,2] = 0.0
x[2,3] = 0.0
d[2] = 30.0
x[3,1] = 10.0
x[3,2] = 20.0
x[3,3] = 0.0
d[3] = 10.0
```