ISyE524: Introduction to Optimization
Problem Set #4
Due: *Tuesday* April 7, at 11:59PM

Notes and Deliverables:

- Submit solutions to all problems in the form of a single PDF file of the IJulia notebook to the course dropbox. (If you do not submit as a single PDF you will lose points.)
- Be sure that your answers (including any pictures you include in the notebook) are in the correct order. (If you do not submit the problems in order, you will lose points.)

1 Portfolio Optimization

We have been given the opportunity to help manage the 524Fund, a portfolio/hedge fund with 1 billion dollars of assets for the next year. We are given the expected value of the return μ and covariance matrix Q of the returns between 225 assets for the next year in which the 524Fund may invest.

The 524Fund currently holds 40 of these assets, and the asset holdings (in percentage of the portfolio) are given in a vector h. The 524Fund has changed management, and we have been given the directive to transition the fund to match/track a new benchmark, the Linde50, whose percentages are given in the vector b.

When fund managers wish to track a benchmark b, they measure of closeness of a portfolio x to the benchmark with "active risk", defined as

$$\sqrt{(x-b)^T Q (x-b)}$$

The current active risk of the 524Fund (holdings h) with respect to the Linde50 (benchmark b) is far too high—almost 33%.

We will write some optimization problems that will reduce the active risk, but do so in a way that minimizes the impact of the transactions.

1-1 Problem Write an optimization problem that will minimize the total number of dollars that must be bought or sold in order to reduce the active risk of the (new) 524Fund holdings to be ≤ 10. (The transaction size is the amount bought + the amount sold) You should use the solver ***Gurobi*** to solve your instance.

hint: You should have (at least) variables for the new holdings in the 524Fund, variables for how much of each asset to buy, and how much of each asset to sell.

Answer:

We will define $N = \{1, 2, \ldots, 225\}$ as the set of assets and the following decision variables:

- x_i: The percentage of the portfolio that is in invested in asset $i \in N$
- b_i: The percentage of the portfolio that we buy of asset $i \in N$

- s_i: The percentage of the portfolio that we sell of asset $i \in N$

Then an optimization model that will minimize the total number of dollars that must be bought or sold in order to reduce the active risk of the (new) 524Fund holdings to be ≤ 10 is the following:

$$\min_{i \in N} (b_i + s_i)$$

$$\begin{aligned} x_i + s_i &= h_i + b_i \quad \forall i \in N \\ \sum_{i \in N} \sum_{j \in N} Q_{ij}(x_i - b_i)(x_j - b_j) &\leq 100 \\ \sum_{i \in N} x_i &= 1 \\ x_i, s_i, b_i &\geq 0 \qquad \forall i \in N \end{aligned}$$

The second constrant could also be written in matrix form as $(x-b)^T Q(x-b) \leq 100$. You could also write the constraint as $\sqrt{(x-b)^T Q(x-b)} \leq 10$.

1-2 Problem Using the data in the given files, `folio_mean.csv`, `folio_cov.csv`, `folio_holding_benchmark.csv`, implement your model from Problem 1-1 to find the minimum total transaction size. You should use the solver ***Gurobi*** to solve your instance.

The code below (also available as `portfolio_market_impact-start.ipynb`) will read the data and also transform the given data for this problem.

Julia Code

```
using DataFrames, CSV, LinearAlgebra, NamedArrays

df = CSV.read("folio_mean.csv", DataFrame, header=false, delim=',')
(n,mmm) = size(df)

# Weekly numbers to annual (and flip returns to make more positive)
mu = -100/7*365*Vector(df[1:n,1])

df2 = CSV.read("folio_cov.csv",DataFrame,header=false,delim=',')

# Weekly numbers to annual, also reduce the risk a bit
Q = 0.5* (100/7*365)^2 * Matrix(df2)

df3 = CSV.read("folio_holding_benchmark.csv", DataFrame, header=false, delim=',')

h = Vector(df3[1:n,1])
b = Vector(df3[1:n,2])

# Current tracking risk
benchmark_return = mu'b

# Current holdings return
holdings_return = mu'*h

```

```
25      # Current tracking risk
26      active_risk = sqrt((h-b)'Q*(h-b))
27
28      println("Benchmark expected return: ", benchmark_return)
29      println("Holdings expected return: ", holdings_return)
30      println("Active Risk: ", active_risk)
```

Be sure to print the following statistics after solving your optimization model:

- The total number of dollars transacted. (You should multiply your objective value by \$1B if you are using portfolio fraction as the decision-variable units)
- The average size of the largest 10 transactions (\$)
- The portfolio expected return (%)
- The portfolio active risk (%)

Answer:

Julia Code

```
1       using JuMP, Gurobi, Statistics, Printf
2       m = Model(Gurobi.Optimizer)
3
4       set_silent(m)
5       @variable(m, 0 <= x[1:n] <= 1) # Units are billions of dollars
6       @variable(m, 0 <= buy[1:n] <= 1)
7       @variable(m, 0 <= sell[1:n] <= 1)
8
9       @expression(m, portfolio_active_variance, (x-b)'Q*(x-b))
10
11      @objective(m, Min, sum(buy[i] + sell[i] for i in 1:n))
12      @constraint(m, [i in 1:n], x[i] == h[i] + buy[i] - sell[i])
13      @constraint(m, portfolio_active_variance <= 100)
14      @constraint(m, sum(x[i] for i in 1:n) == 1)
15
16      optimize!(m)
17      xsol = value.(x)
18      buysol = value.(buy)
19      sellsol = value.(sell)
20
21      REFSIZE = 1.0e9
22
23      # Put the two vectors together, then sorts them big to small
24      transactions = sort([buysol; sellsol],rev=true)
25      avg10 = REFSIZE*mean(transactions[1:10])
26
27      dollars_transact = REFSIZE*objective_value(m)
28
29      expected_return = mu'*xsol
```

```
    active_risk = sqrt(value(portfolio_active_variance))

    # buy_ix = findall(buysol .> 1e-6)
    # sell_ix= findall(sellsol .> 1e-6)

    # println("Buy: ", length(buy_ix), " different stocks and sell ", length(sell_ix), " different stocks")
    # println("Max buy is \$", 1.0e+9*maximum(buysol))
    # println("Max sell is \$", 1.0e+9*maximum(sellsol))

    @printf("Dollars Transacted (\$): %8.4e\n", dollars_transact)
    @printf("Average size of 10 largest transactions (\$): %8.4e\n", avg10)
    println("Expected Return (%): ", round(expected_return, digits=4))
    println("Active Risk (%): ", round(active_risk, digits=4))
```

Gives

Julia Answer

```
Dollars Transacted ($): 1.0931e+09
Average size of 10 largest transactions ($): 4.9732e+07
Expected Return (%): 8.9753
Active Risk (%): 10.0
```

1-3 Problem The 524Fund management has said that your transaction sizes from Problem 1-2 are too large—trades of that size would negatively impact the prices of the assets—and they would like for you to plan a different set of trades with less market impact.

Write an optimization model that minimizes the total *market impact* of the trades necessary to get the active risk to below 10%. The 524Fund manages total market impact as

$$\sum_{i=1}^{n} y_i^{3/2},$$

where y_i is the size of the transaction for asset i (again, the size of the transaction is the amount bought + amount sold).

You have been instructed to model this using the epigraph of the individual nonlinear functions:

$$\sum_{i=1}^{n} t_i,$$

where $t_i \geq y_i^{3/2}$ $\forall i = 1, 2, \ldots n$, and rotated second order cone constraints. Be sure to clearly define your new decision variables, including any auxiliary variables you need to model the cone constraints.

Answer:

We will use the decision variables x_i, b_i, s_i defined in the answer to Problem 1-1, but also define the following decision variables:

- y_i The size of the transaction (in %) of asset $i \in N$. $(y_i = b_i + s_i)$

- t_i: An epigraph variable for modeling $t_i \geq y_i^{3/2}$
- u_i: An auxiliary variable required for modeling the two rotated second order cone constraints required to do the rational power

$$\begin{aligned}
&\min_{i \in N}(t_i) \\
x_i + s_i &= h_i + b_i \quad \forall i \in N \\
y_i &= s_i + b_i \quad \forall i \in N \\
\sum_{i \in N} \sum_{j \in N} Q_{ij}(x_i - b_i)(x_j - b_j) &\leq 100 \\
\sum_{i \in N} x_i &= 1 \\
2u_i t_i &\geq y_i^2 \quad \forall i \in N \\
\frac{1}{4} y_i &\geq u_i^2 \quad \forall i \in N \\
x_i, s_i, b_i &\geq 0 \quad \forall i \in N
\end{aligned}$$

1-4 Problem Implement your model from Problem 1-3 to find the new suggested set of holdings. Print the same set of statistics as Problem 1-2.

- The total number of dollars transacted. (You should multiply your objective value by \$1B)
- The average size of the largest 10 transactions (\$)
- The portfolio expected return (%)
- The portfolio active risk (%)

Answer:

Julia Code

```
# Now want to minimize the total market impact

using JuMP, Gurobi, Statistics, Printf
m = Model(Gurobi.Optimizer)
set_silent(m)

@variable(m, 0 <= x[1:n] <= 1) # Units are billions of dollars
@variable(m, 0 <= buy[1:n] <= 1)
@variable(m, 0 <= sell[1:n] <= 1)
@variable(m, 0 <= transact[1:n])
@variable(m, 0 <= u[1:n])
@variable(m, 0 <= t[1:n])

@expression(m, portfolio_active_variance, (x-b)'Q*(x-b))
```

```
@objective(m, Min, sum(t[i] for i in 1:n))
@constraint(m, [i in 1:n], x[i] == h[i] + buy[i] - sell[i])
@constraint(m, [i in 1:n], transact[i] == (buy[i] + sell[i]))
@constraint(m, portfolio_active_variance <= 100)
@constraint(m, sum(x[i] for i in 1:n) == 1)

# t_i >= transact_i^(3/2)

# 2 u_i t_i >= transact_i^2
@constraint(m, [i in 1:n], [u[i], t[i], transact[i]] in RotatedSecondOrderCone())

# 1/4 transact_i >= u_i^2
@constraint(m, [i in 1:n], [0.125, transact[i], u[i]] in RotatedSecondOrderCone())

optimize!(m)
xsol = value.(x)
buysol = value.(buy)
sellsol = value.(sell)

REFSIZE = 1.0e9

# Put the two vectors together, then sorts them big to small
transactions = sort([buysol; sellsol],rev=true)
avg10 = REFSIZE*mean(transactions[1:10])

dollars_transact = REFSIZE*(sum(buysol[i] + sellsol[i] for i in 1:n))

expected_return = mu'*xsol
active_risk = sqrt(value(portfolio_active_variance))

buy_ix = findall(buysol .> 1e-6)
sell_ix= findall(sellsol .> 1e-6)

# println("Buy: ", length(buy_ix), " different stocks and sell ", length(sell_ix), " different stocks")
# println("Max buy is \$", 1.0e+9*maximum(buysol))
# println("Max sell is \$", 1.0e+9*maximum(sellsol))

@printf("Dollars Transacted (\$): %8.4e\n", dollars_transact)
@printf("Average size of 10 largest transactions (\$): %8.4e\n", avg10)
println("Expected Return (%): ", round(expected_return, digits=4))
println("Active Risk (%): ", round(active_risk, digits=4))
```

Julia Answer

```
Dollars Transacted ($): 1.1650e+09
Average size of 10 largest transactions ($): 4.5922e+07
Expected Return (%): 8.8552
Active Risk (%): 9.9981
```

2 Making An Indefinite Matrix Definite

The following matrix is indefinite:

$$Q = \begin{pmatrix} 0 & 0 & -2 & -4 & 0 & 1 \\ 0 & 1 & -1 & -1 & 3 & -4 \\ -2 & -1 & -1 & -5 & 7 & -4 \\ -4 & -1 & -5 & -3 & 7 & -2 \\ 0 & 3 & 7 & 7 & -1 & -2 \\ 1 & -4 & -4 & -2 & -2 & 0 \end{pmatrix}$$

2-1 Problem Use Julia to compute the eigenvalues of Q and print them out. You will need to use the LinearAlgebra package, and then `eigen(Q)` returns the eigenvalues and eigenvectors of Q

Answer:

Julia Code

```
using LinearAlgebra, Printf

Q = [
0 0 -2 -4 0 1;
0 1 -1 -1 3 -4;
-2 -1 -1 -5 7 -4;
-4 -1 -5 -3 7 -2;
0 3 7 7 -1 -2;
1 -4 -4 -2 -2 0
]

c = [-1, 0, 2, -2, 4, 0]

eigs = eigen(Q)
n = length(c)
for i in 1:n
    @printf "Eig[%d] = %.4f\n" i eigs.values[i]
end
```

Gives

Julia Answer

```
Eig[1] = -16.1191
Eig[2] = -3.7566
Eig[3] = -0.5923
Eig[4] = 2.2331
Eig[5] = 3.8457
Eig[6] = 10.3892
```

2-2 Problem It is well-known that if a real symmetric matrix $Q \prec 0$ with minimum eigenvalue $\lambda_1 < 0$, then the matrix obtained by subtracting λ_1 from the diagonal is PSD, i.e. $Q - \lambda_1 I \succeq 0$. Thus if we add a *total* of $6 \times |\lambda_1|$ to the diagonal of Q, the results matrix is PSD.

But we could add *less* to make the matrix PSD. Write a semidefinite program that will find a ***non-negative*** diagonal matrix D such that $Q + D \succeq 0$ and $\sum_{i=1}^{6} D_{ii}$ is minimized?

Answer:

We will have a (symmetric) matrix of variables $X \in \mathbb{R}^{6\times 6}$. Then a semidefinite program that will find a ***non-negative*** diagonal matrix D such that $Q+D \succeq 0$ and $\sum_{i=1}^{6} D_{ii}$ is minimized is the following:

$$\min \sum_{j=1}^{6} X_{jj}$$

$$\begin{aligned} X + Q &\succeq 0 \\ X_{ij} &= 0 \quad \forall i = 1\ldots, 6, j = 1, \ldots 6, i \neq j \\ X_{ij} &\geq 0 \quad \forall i = 1\ldots, 6, j = 1, \ldots 6 \end{aligned}$$

The objective here could also be written as teh trace of X, $\mathrm{tr}(X)$. I think some students also write this using the diag operator. Here, let $d \in \mathbb{R}^6_+$ be the decision variables, then you can write the answer as

$$\min \sum j = 1^6 d_j$$

$$\begin{aligned} \mathrm{diag}(d) + Q &\succeq 0 \\ d_j &\geq 0 \quad \forall j = 1, \ldots, 6 \end{aligned}$$

2-3 Problem Implement your SDP from Problem 2-2 in Julia and JuMP. Solve the resulting SDP with the SCS solver. Print the (diagonal) entries of your matrix D and their sum.

Answer:

Julia Code

```
# Now lets find the smallest amount to Q to make it PSD
using JuMP, SCS
model = Model(SCS.Optimizer)
@variable(model, X[1:n, 1:n] >= 0)
@objective(model, Min, sum(X[j,j] for j in 1:n))

for i in 1:n
    for j in 1:n
        if i != j
            @constraint(model, X[i,j] == 0)
        end
    end
end

@constraint(model, (X + Q) in PSDCone())

```

```
    #print(model)

    optimize!(model)
    @assert is_solved_and_feasible(model)
    #print(objective_value(model))
    Xval = value.(X)
    println("Objective Value: ", round(objective_value(model), digits=3))
    println("Diagonal Elements: ")
    for j=1:6
        println("D[", j, ",", j, "] = ", round(Xval[j,j], digits=3))
    end
```

Gives

Julia Answer

```
Objective Value: 78.0
Diagonal Elements:
D[1,1] = 5.0
D[2,2] = 8.0
D[3,3] = 20.0
D[4,4] = 22.0
D[5,5] = 16.0
D[6,6] = 7.0
```

2-4 Problem Now we would like to find a (not-necessarily diagonal) matrix X such that the matrix $Q + X$ is positive-semidefinite, and the total (absolute-value) of the sum of the entries of X is as small as possible. Write a semidefinite program to find a symmetric matrix X such that $\sum_{i=1}^{6}\sum_{j=1}^{6}|X_{ij}|$ is as small as possible, and $Q + X \succeq 0$. Note here that unlike problem 2-2, we are not requiring the individual elements of our matrix X to be non-negative, but we want to minimize the sum of their absolute values.

Answer:
Here we will need three (6×6) symmetric matrices of variables

- X_{ij}: Matrix element (i, j) to add to Q_{ij}
- P_{ij}: Positive part of X_{ij}
- P_{ij}: Negative part of X_{ij}

$$\min \sum_{i=1}^{6}\sum_{j=1}^{6}(P_{ij} + N_{ij})$$

$$\begin{aligned} X + Q &\succeq 0 \\ X_{ij} &= P_{ij} - N_{ij} \quad \forall i = 1\ldots, 6, j = 1, \ldots 6 \\ P_{ij}, N_{ij} &\geq 0 \quad \forall i = 1\ldots, 6, j = 1, \ldots 6 \end{aligned}$$

2-5 Problem Implement your model from Problem (2-4) in Julia and JuMP. Print your matrix X and the sum of the absolute values of its entries.

Answer:

Julia Answer

```
Absolute value of matrix entries: 78.0

6×6 Matrix{Float64}:
 2.73361    0.441088    0.388126   0.328122   -0.375837    0.734781
 0.441089   7.08547     0.197721   0.0789439  -0.0855225   0.110722
 0.388126   0.197721   13.1811     3.13256    -2.83215     0.268113
 0.328122   0.0789438   3.13256   12.5465     -5.76302     0.151187
-0.375837  -0.0855223  -2.83215   -5.76302     6.81257    -0.130467
 0.734782   0.110722    0.268113   0.151187   -0.130467    5.60403
```

3 Checked Luggage

Suppose that you trying to pack as many souvenirs as possible to bring home from your trip, but your suitcase has a limited capacity. It can hold a maximum of 30 pounds of weight and 15 gallons of volume. The weights and volumes are as follows:

Souvenir number	1	2	3	4	5	6	7	8	9	10
Weight (lbs)	5	6	7	6	4	6	7	3	8	5
Volume (gal)	2	4	5	3	3	2	3	1	2	4

3-1 Problem Write an integer programming model to determine which souvenirs to pack. (Your goal is to just maximize the total number of souvenirs).

Answer:
Let $x_i \in \{0,1\}, i = 1,\ldots,10$ be a collection of binary variables indicating whether or not souvenir number i is packed. Then an IP model to determine a maximum number to pack while obeying weight and volume restrictions is the following:

$$\max \sum_{i=1}^{10} x_i$$

$$\begin{aligned} \text{s.t.} \quad & 5x_1 + 6x_2 + 7x_3 + 6x_4 + 4x_5 + 6x_6 + 7x_7 + 3x_8 + 8x_9 + 5x_{10} \leq 30 && \text{(Weight)} \\ & 2x_1 + 4x_2 + 5x_3 + 3x_4 + 3x_5 + 2x_6 + 3x_7 + 1x_8 + 2x_9 + 4x_{10} \leq 15 && \text{(Volume)} \\ & x_i \in \{0,1\} \ \forall i = 1,\ldots,10 \end{aligned}$$

3-2 Problem Implement your model from Problem 3-1 in JuMP and display the selected items.

Answer:

Julia Code

```
    # problem data (we'll use an array here)
weights = [ 5, 6, 7, 6, 4, 6, 7, 3, 8, 5 ]
volumes = [ 2, 4, 5, 3, 3, 2, 3, 1, 2, 4 ]
n = length(volumes) # total number of souvenirs

using JuMP, HiGHS
m = Model(HiGHS.Optimizer)

@variable(m, x[1:n], Bin)
@constraint(m, sum( weights[i] * x[i] for i=1:n ) <= 30 )
@constraint(m, sum( volumes[i] * x[i] for i=1:n ) <= 15 )
@objective(m, Max, sum(x) )
optimize!(m)

println(" binary solution vector is ",value.(x))
println("Can take ", objective_value(m), " items")
```

Gives the following solution:

Julia Answer

```
binary solution vector is
[1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0]
Can take 6.0 items
```

3-3 Problem Now suppose that you have a very demanding partner, and they are not happy with the answer obtained when you have just 30 pounds max of weight and 15 gallons of volume. They would like to know *all* their options. That is, for each possible (integer) value of (weight, volume) pairs, how many items can you pack? You can get it to look fancy if you use the code below, and then your partner will be very impressed!

(Make an array of size #volumes, and columns #weights, and then fill in the matrix entry with the value you would like to plot (the objective value)).

You should use the *Gurobi* solver for this problem. (We had seen some bugs in HiGHS).

This code will make a 3D bar plot of the array in vals (using PyPlot)

Julia Code

```
using PyPlot

function plot3d(vals)
    nrows = size(vals,1)
    ncols = size(vals,2)
    _x = 1:nrows
    _y = 1:ncols
    # Make a meshgrid
```

```
        x = _x' .* ones(ncols)
        y = ones(nrows)' .* _y
        # Unravel
        x = vec(x)
        y = vec(y)
        # Heights
        dz = vec(vals')
        z = zeros(length(dz))

        dx = 0.4
        dy = 0.4

        bar3D(x,y,z,dx,dy,dz)
        # Only needed in vscode
        display(gcf())
    end
    ;
```

Here is code in Julia's `Plots` package.

Julia Code

```
    import Plots
    Plots.gr()

    xs = collect(1:w_max)
    ys = collect(1:v_max)

    X = [x for x = xs for _ = ys]
    Y = [y for _ = xs for y = ys]

    Plots.gui()
    Plots.surface(X, Y, vec(ns'))
```

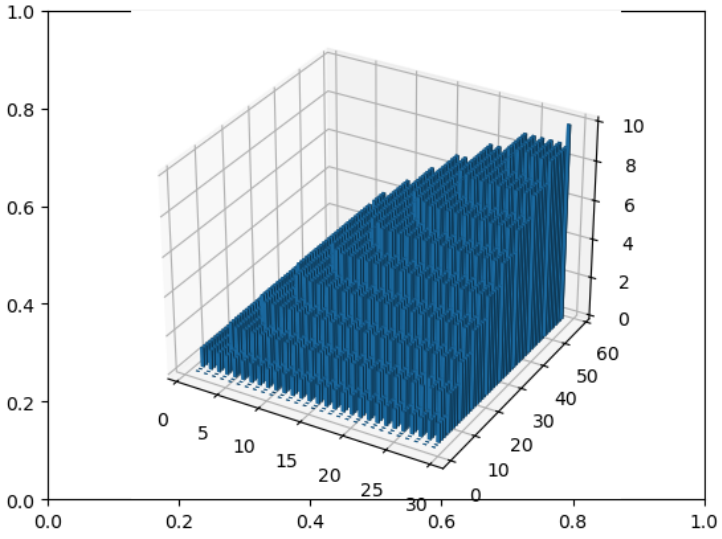
Answer:

Here is my code:

Julia Code

```
using JuMP, HiGHS

# Here we will just solve it in a loop for every possible value
max_weight = sum(weights[i] for i=1:n)
max_volume = sum(volumes[i] for i=1:n)
num_items = zeros(max_volume, max_weight)

for v = 1:max_volume
    for w = 1:max_weight
        m = Model(HiGHS.Optimizer)
        set_silent(m)

```

```
            @variable(m, x[1:n], Bin)
            @constraint(m, sum( weights[i] * x[i] for i=1:n ) <= w )
            @constraint(m, sum( volumes[i] * x[i] for i=1:n ) <= v )
            @objective(m, Max, sum(x) )
            optimize!(m)
            num_items[v,w] = objective_value(m)
        end
    end

    plot3d(num_items)
```



4 Steam Power

A power plant has three boilers. If a given boiler is operated, it can be used to produce a quantity of steam (in tons) between the minimum and maximum given in the table below. This table also gives the cost and carbon emissions per ton of steam produced from each boiler.

Boiler Number	Min Steam (Tons)	Max Steam (Tons)	Cost/Ton ($)	Carbon/ton (lbs)
B1	400	1000	10	20
B2	200	900	8	30
B3	300	800	7	35

Steam from the boilers is used to produce power on three turbines. If operated, each turbine can process an amount of steam (in tons) between the minimum and maximum given in the table below. The table also gives the amount of power (in KwH) produced and the procesing cost per ton of steam processed for each turbine. The power plant must produce 8000 KwH of power.

Turbine	Min (tons)	Max (tons)	KwH per Ton of Steam	Processing Cost per Ton ($)
T1	300	600	4	2
T2	500	800	5	3
T3	600	900	6	4

4-1 Problem Formulate an integer program to help the power plant determine how to meet the requirements at minimum cost.

Answer:

Decision Variables:

- x_i: Amount of steam to produce from boiler $i = 1, 2, 3$
- t_j: Amount of steam to send through turbine $j = 1, 2, 3$
- y_i: 1 if operate boiler i, 0 otherwise
- z_j: 1 if operate turbine j, 0 otherwise

Objective:

$$\min 10x_1 + 8x_2 + 7x_3 + 2t_1 + 3t_2 + 4t_3$$

Constraints:

Produce 8,000 kWh of electricity

$$4t_1 + 5t_2 + 6t_3 \geq 8,000$$

Boilers must operate within their limits when on, and be zero otherwise:

$$400y_1 \leq x_1 \leq 1,000y_1$$

$$200y_2 \leq x_2 \leq 900y_2$$

$$300y_3 \leq x_3 \leq 800y_3$$

Turbines must operate within their limits when on, and be zero otherwise:

$$300z_1 \leq t_1 \leq 600z_1$$

$$500z_2 \leq t_2 \leq 800z_2$$

$$600z_3 \leq t_3 \leq 900z_3$$

Turbines must get their steam from somewhere

$$t_1 + t_2 + t_3 \leq x_1 + x_2 + x_3$$

Finally, we write the sign and integer requirements on the variables:

$$
\begin{aligned}
x_i &\geq 0 \quad i = 1, 2, 3 \\
t_j &\geq 0 \quad j = 1, 2, 3 \\
y_i &\text{ binary}, \quad i = 1, 2, 3 \\
z_j &\text{ binary}, \quad j = 1, 2, 3
\end{aligned}
$$

4-2 Problem Implement your model from Problem 4-1 in Julia and JuMP to determine the optimal way to meet power plant demand.

Answer:

Julia Code

```
# Sets
B = [Symbol("Boiler",i) for i in 1:3]
T = [Symbol("Turbine",i) for i in 1:3]

# Parameters
min_steam = Dict(zip(B,[400,200,300]))
max_steam = Dict(zip(B,[1000,900,800]))
steam_cost = Dict(zip(B,[10,8,7]))
carbon_per_ton = Dict(zip(B,[20,30,35]))

min_turbine = Dict(zip(T,[300,500,600]))
max_turbine = Dict(zip(T,[600,800,900]))
kwh_per_ton = Dict(zip(T,[4,5,6]))
turbine_cost = Dict(zip(T,[2,3,4]))

min_power = 8000

using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
set_silent(m)

# Variables
# Tons of steam from boiler i
@variable(m, x[B] >= 0)
# Tons of steam through turbine j
@variable(m, t[T] >= 0)
# 1 if operate boiler i, 0 otherwise
@variable(m, y[B], Bin)
# 1 if operate turbine i, 0 otherwise
@variable(m, z[T], Bin)

#Min cost
@objective(m, Min, sum(steam_cost[i]*x[i] for i in B) +
    sum(turbine_cost[i]*t[i] for i in T))

# Max 8000 kKW of electricity
```

```
@constraint(m, sum(kwh_per_ton[i]*t[i] for i in T) >= min_power)

# Operate boilers within limits
@constraint(m, [i in B], min_steam[i]*y[i] <= x[i])
@constraint(m, [i in B], x[i] <= max_steam[i]*y[i])

# Operate turbines within limits
@constraint(m, [i in T], min_turbine[i]*z[i] <= t[i])
@constraint(m, [i in T], t[i] <= max_turbine[i]*z[i])

# Make sure turbine tons don't exceed boiler tons
@constraint(m, sum(x[i] for i in B) >= sum(t[i] for i in T))

optimize!(m)
println("Minimum cost is ", objective_value(m))
for i in B
    if value(x[i]) > 0.01
        println(value(x[i]), " tons from ", i)
    end
end
for i in T
    if value(t[i]) > 0.01
        println(value(t[i]), " tons through ", i)
    end
end

```

Julia Answer

```
Minimum cost is 15720.0
620.0 tons from Boiler2
800.0 tons from Boiler3
520.0 tons through Turbine2
900.0 tons through Turbine3
```

5 524 Cheese

The 524 Cheese Company (524CC) produces two types of cheese: Colby and Mozzarella. Currently, 524CC has 120 pounds of Colby and 80 pounds of Mozzarella in stock.

Processing a pound of milk costs \$0.20 and yields 0.5 pounds of Colby cheese and 0.4 pounds of Mozzarella. Either type of cheese can be stored from week to week, but milk cannot be stored. It costs \$0.25 per pound of cheese (either type) that is stored from one week to the next, and at most 500 pounds (total) can be stored at any time. Colby cheese is sold for \$2.50 per pound, and Mozzarella cheese is sold for \$3 per pound.

The price of milk is a complicated function of future (per unit) prices and delivery costs. There is a fixed cost of f_t if delivery of milk occurs in week t, and the per unit price of milk in week t is p_t

The table below shows the maximum amount of each type of cheese 524CC could sell in the next 8 weeks as well as the fixed and variable costs for milk. Demand that is not met in the week it occurs

cannot be carried over to a later week (i.e., no backlogging is allowed).

Week	Colby	Mozzarella	f_t	p_t
1	150	200	1000	1
2	200	400	1400	0.8
3	225	300	800	0.8
4	50	500	1200	1.2
5	400	100	600	1.2
6	50	500	1000	1.0
7	300	200	400	1.5
8	200	350	800	0.6

5-1 Problem Write an integer programming model that maximizes the 524CC profit over the planning horizon.

Hint: Your model should probably have (at least) the following decision variables:

- Pounds of milk to buy each week
- Pounds of colby to sell each week
- Pounds of mozerrella to sell each week
- Inventory of colby at end of each each
- Inventory of mozerelaa at end of each week
- Indicator if whether or not milk was bought each week

Answer:

Let $T = \{1, 2, \ldots, 8\}$, and define the following decision variables:

- x_t: Pounds of milk to buy in week $t \in T$
- y_t: Pounds of colby to sell in week $t \in T$
- z_t: Pounds of mozerella to sell in week $t \in T$
- I_t: Pounds of colby in inventory at end of to sell in week $t \in T$
- J_t: Pounds of mozerella in inventory at end of to sell in week $t \in T$
- w_t: A binary indicator vector for whether or not 524CC purchases milk in week $t \in T$

To make life easier, I will also define the parameters d_t^{col} and d_t^{moz} to be the maximum amount of colby and mozzerella, respectively, than can be sold in week $t \in T$. Finally, I define a parameter M_t, which is the maximum amount of milk that can be purchased in week $t \in T$. Then an integer programming model that maximizes the 524CC profit over the planning horizon is the following:

$$\max \sum_{t \in T} (2.5y_t + 3.0z_t) - \sum_{t \in T} (0.2x_t + 0.25(I_t + J_t) + f_t w_t + p_t x_t)$$

$$\begin{aligned}
120 + 0.5x_1 &= I_1 + y_1 \\
I_{t-1} + 0.5x_t &= I_t + y_t \quad \forall t \in \{2, \ldots, 8\} \\
80 + 0.4x_1 &= J_1 + z_1 \\
J_{t-1} + 0.4x_t &= J_t + Z_t \quad \forall t \in \{2, \ldots, 8\} \\
I_t + J_t &\leq 500 \quad \forall t \in T \\
x_t &\leq M_t w_t \quad \forall t \in T \\
y_t &\leq d_t^{\mathsf{col}} \quad \forall t \in T \\
z_t &\leq d_t^{\mathsf{moz}} \quad \forall t \in T \\
x_t, y_t, z_t, I_t, J_t &\geq 0 \quad \forall t \in T \\
w_t &\in \{0, 1\} \quad \forall t \in T
\end{aligned}$$

5-2 Problem Implement your model from Problem 5-1 in JuMP to find the optimal purchase and production schedule for 524CC.

Print out the following information:

- Total maximum profit
- The weeks that milk is purchased
- The total inventory of cheese at the end of each week.

You can use the code here to load the data for the problem. (Also available on Canvas)

Julia Code

```
# initial colby (pounds)
i_colby = 120

# initial mozerella (pounds)
i_moz = 80

# Demand for colby cheese (pounds)
d_colby = [150 200 225 50 400 50 300 200]
# Demand for mozerella cheese (pounds)
d_moz = [200 400 300 500 100 500 200 350]

# Fixed cost for a delivery of milk ($)
fc_milk = [1000 1400 800 1200 600 1000 400 800]

# Per-unit cost for milk ($/pound)
p_milk = [1 0.8 0.8 1.2 1.2 1.0 1.5 0.6]

# Processing cost of milk ($/pound)
milk_proc_cost = 0.2
```

```
20
21      # inventory cost ($/pound)
22      cheese_inventory_cost = 0.25
23
24      # max total inventory (pounds)
25      max_inventory = 500
26
27      # colby price ($/pound)
28      colby_price  = 2.5
29
30      # mozzerella price ($/pound)
31      moz_price = 3.0
32
33      # colby/milk
34      colby_per_milk = 0.5
35
36      # moz/milk
37      moz_per_milk = 0.4
38
39      # Number of time periods
40      T = 8
41
42      # The maximum milk you would buy in any period is surely no more than the total demand for cheese
43      max_milk = [sum(d_colby[1:t])+sum(d_moz[1:t]) for t in 1:T]
44
45      using JuMP, HiGHS
46      m = Model(HiGHS.Optimizer)
47
48      # Pounds of milk to buy
49      @variable(m, x[1:T] >= 0)
50
51      # Pounds of colby to sell
52      @variable(m, 0 <= y[t in 1:T] <= d_colby[t])
53
54      # Pounds of moz to sell
55      @variable(m, 0 <= z[t in 1:T] <= d_moz[t])
56
57      # Inventory of Colby (at end of t)
58      @variable(m, I[0:T] >= 0)
59      @constraint(m, I[0] == i_colby)
60
61      # Inventory of Moz (at end of t)
62      @variable(m, J[0:T] >= 0)
63      @constraint(m, J[0] == i_moz)
64
65      # Did you buy milk this week?
66      @variable(m, w[1:T], Bin)
67
68
69      @expression(m, revenue, sum(colby_price*y[t] + moz_price*z[t] for t in 1:T))
70
71      @expression(m, cost, sum(milk_proc_cost*x[t] + cheese_inventory_cost*(I[t]+J[t]) + fc_milk[t]*w[t] + p_milk
72
```

```
    @objective(m, Max, revenue - cost)

    # Colby balance
    @constraint(m, Balance_Colby[t in 1:T], I[t-1] + colby_per_milk*x[t] == I[t] + y[t])

    # Moz balance
    @constraint(m, Balance_Moz[t in 1:T], J[t-1] + moz_per_milk*x[t] == J[t] + z[t])

    # Max inventory
    @constraint(m, Max_Inv[t in 1:T], I[t] + J[t] <= max_inventory)

    # Fixed cost
    @constraint(m, Fixed_Charge[t in 1:T], x[t] <= max_milk[t]*w[t])


    optimize!(m)

    xvals = value.(x)
    yvals = value.(y)
    zvals = value.(z)

    Ivals = value.(I)
    Jvals = value.(J)
    wvals = value.(w)

    # Compute the required statistics to print
    max_profit = objective_value(m)
    weeks_purchased = [i for i in 1:T if wvals[i] > 0.1 ]
    total_inventory = [Ivals[i]+Jvals[i] for i in 1:T]

    println("Profit: (\$)", round(max_profit,digits=2))

    println("Weeks milk purchased: ", weeks_purchased)

    println("Total cheese inventory at end of week: ", total_inventory)

```

Gives

Julia Answer

```
Profit: ($)1442.78
Weeks milk purchased: [3, 7]
Total cheese inventory at end of week: [0.0, 0.0, 500.0, 294.44, -0.0, 0.0, 400.0, 0.0]
```