ISyE524: Introduction to Optimization
Problem Set #1
Due: February 11, 2024 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 need to write answers on paper, you can take a picture of you work and add that to your IJulia notebook

1 Graphical Method Extravaganza

Consider the following linear programming problem:

$$\begin{aligned} \max\ & -4x_1 + 2x_2 \\ \text{subject to:}\quad & x_2 \le 5 \\ & 2x_1 - x_2 \ge 2 \\ & x_1,\ x_2 \ge 0 \end{aligned}$$

1-1 Problem Find an optimal solution to this linear program using the graphical solution method. (Show the feasible region as part of your answer.)

1-2 Problem How many optimal solutions does this linear program have? If more than one, provide another optimal solution.

1-3 Problem Now suppose the objective is changed to maximize $-x_1 + x_2$. Answer questions 1-1 and 1-2 again.

1-4 Problem Now suppose the objective is changed to minimize $-x_1 + x_2$. Does the LP now have an optimal solution? If so, provide one. If not, what do we call this LP?

2 Standard Form

Consider the following LP:

$$\begin{aligned} \min\ & 2x_1 - x_2 + 3x_3 - 2x_4 \\ \text{s.t. } & 3x_2 - x_3 + 4x_4 \ge 10 \\ & 4x_1 - 7x_2 + x_3 - x_4 = 5 \\ & x_1 - 3x_3 + 8x_4 \le 3 \\ & x_1, x_4 \ge 0 \\ & x_2, x_3 \text{ Unrestricted in Sign (Free)} \end{aligned}$$

2-1 Problem Implement the instance and Julia/JuMP and solve it, printing the optimal solution and optimal solution value.

2-2 Problem Convert the problem to the standard form:

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

What are A, b, c,and x in this problem? Clearly indicate how the decision variables of your transformed LP relate to those of the original LP.

2-3 Problem Solve the standard-form LP in Julia and report the objective value and the value of each decision variable in an optimal solution to the original LP.

3 Mmmmmmmmmmmmmmmm. Pork.

Jacob's Meat Packing Plant produces 480 hams, 400 pork bellies, and 230 picnic hams every day. Each of these products can be sold either fresh or smoked. The total number of hams, bellies, and picnics that can be smoked during a normal working day is 420. In addition, up to 250 products can be smoked on overtime, at a higher cost (and therefore a lower net profit). The net profits (in $ per product) are the following:

	Fresh	Smoked (Regular)	Smoked (Overtime)
Hams	8	14	11
Bellies	4	12	7
Picnics	4	13	9

3-1 Problem Write a *linear* program that will maximize the total net profit.

3-2 Problem Create your instance from Problem 3-1 in the JuMP modeling language and solve it, printing the profit and number of each type of meat product to produce.

4 Mission Impossible

```
Lisa:  ''I'm going to become a vegetarian''
Homer:  ''Does that mean you're not going to eat any pork?''
Lisa:  ''Yes''
Homer:  ''Bacon?''
Lisa:  ''Yes Dad''
Homer:  ''Ham?''
Lisa:  ''Dad all those meats come from the same animal!''
Homer:  ''Right Lisa, some wonderful, magical animal!''
```

Jacob's Meat Alternative Plant (`jmap`) produces a set of plant-based pork products P using Impossible Meat. (For example, it may be that $P = \{\text{Ham, Bacon, Pork Belly}\}$.) A product $p \in P$ can be prepared (and then sold) using a cooking method $c \in C$. (For example, it may be that $C = \{\text{Fresh, Smoked, Fried}\}$.) `jmap` produces b_p products of type $p \in P$ per day. The total number of products that can be prepared using cooking method $c \in C$ each day is given by the parameter u_c. The net revenue to `jmap` of product $p \in P$ when prepared by cooking method $c \in C$ is h_{pc}. Each cooking method $c \in C$ has an associated cost of q_c for any product sold using that method.

4-1 Problem Write a linear program that will maximize the total net profit for `jmap`. Just write the mathematical model using general notation.

4-2 Problem Implement your instance from Problem 4-1 in JuMP on the data in the data file `pork-general.csv`. You can either re-type the data, or use Julia DataFrames and CSV to build.

If you wish, you can use this code to help read the data from the CSV file:

Julia Code

```
# You may need to Pkg.add() these
using DataFrames, CSV, NamedArrays
df = CSV.read("pork-general.csv", DataFrame, delim =',')

cook_type = propertynames(df)[3:end]
product = convert(Array, df[3:end,1])

max_daily_product = Dict(zip(product,df[3:end,2]))
max_daily_cook = Dict(zip(cook_type,df[1,3:end]))
proc_cost = Dict(zip(cook_type,df[2,3:end]))

h = Matrix(df[3:end,3:end])
h_NA = NamedArray(h, (product, cook_type), ("Product", "Cook Type")) ;
```

5 Optiyours and Optimine

A mining company ("OptiMine") is planning mine operations for the next week. There is a set I of mine locations available for ore extraction. Each mine location $i \in I$ has u_i tons of ore available. Mining at each location $i \in I$ has an associated cost per ton c_i. Finally, there is a set of elements E, and each mine location $i \in I$ has a different percentage of each type of element. The value α_{ei} is the percentage of element $e \in I$ in the ore mined from location $i \in I$. The Optimine company must mine at least K tons of ore total, and each element $e \in E$ must have at least a percentage β_e.

5-1 Problem Formulate a linear program to help OptiMine plan mining operations to meet requriments at a minimal cost. Give a general form (parameters only – no numbers) of the math model.

5-2 Problem Here is ChaptGPT (v4.0's) answer for this problem, written in Julia and JuMP:

Julia Code

```
    using JuMP
    using GLPK # Or any other LP solver
```

```

    # Define the data structures
    I = # Set of mine locations
    E = # Set of elements
    u = # Dictionary mapping each location i ∈ I to the available tons of ore u_i
    c = # Dictionary mapping each location i ∈ I to the cost per ton c_i
    α = # Dictionary mapping each (element, location) pair (e, i) to the percentage α_{ei}
    β = # Dictionary mapping each element e ∈ E to the required minimum percentage β_e
    K = # Minimum total tons of ore to be mined

    # Create a new model
    model = Model(GLPK.Optimizer)

    # Decision variables
    @variable(model, x[i in I] >= 0) # Tons of ore to mine from each location

    # Objective: Minimize total cost
    @objective(model, Min, sum(c[i] * x[i] for i in I))

    # Constraint: Mine at least K tons of ore in total
    @constraint(model, sum(x[i] for i in I) >= K)

    # Constraints: Ensure the percentage of each element
    for e in E
        @constraint(model, sum(α[(e, i)] * x[i] for i in I) / sum(x[i] for i in I) >= β[e])
    end

    # Solve the model
    optimize!(model)

    # Check results
    if termination_status(model) == MOI.OPTIMAL
        println("Optimal solution found")
        for i in I
            println("Mine ", i, " - Tons of ore: ", value(x[i]))
        end
    else
        println("No optimal solution found")
    end
```

Is this a correct linear programming model for the problem? If it is not correct, please explain why and fix the model. If it is a correct model, please comment on any differences between your mathematical model and ChatGPT (v4.0)'s Julia/JuMP implementation.

5-3 Problem Implement and solve this instance of the model in Julia/JuMP. Display the optimal objective value and the optimal variable values.

The cost, ore available, tons of each element required, and percentage of elements in the ore at each mining location are given in the .csv file "mine.csv." A sample of the data is given in the table below. There is a minimum percentage of each element required across the total ore extracted, also given in the .csv file.

The data is provided in the "optimine.csv" on Canvas. You can use the code snippet provided below to read the .csv files and load them into Julia.

Julia Code

```
#You might need to run "Pkg.add(...)" before using these packages
using DataFrames, CSV, NamedArrays

#Load the CSV data file (should be in same directory as notebook)
df = CSV.read("optimine.csv",DataFrame,delim=',');

# create a list of mines
mines = convert(Array,df[2:end,1])

# create a list of elements
#  here we take from the DataFrame header (into Julia Symbol)
elements = propertynames(df)[2:6]

# create a dictionary of the total cost of mining at each location
c = Dict(zip(mines,df[2:end,7]))

# create a dictionary of the max tons available at each location
u = Dict(zip(mines,df[2:end,8]))

# create a dictionary of the amount required of each element
β = Dict(zip(elements,df[1,2:6]))

# create a matrix of the % of each element at each loation
mine_element_matrix = Matrix(df[2:end,2:6])

# rows are mines, columns are elements
α = NamedArray(mine_element_matrix, (mines, elements), ("mines","elements"))
```