Session 22: Lab 2

Learning Objectives:

- Use LP to analyze a realistic business case in supply chain management. (Analyze)
- Formulate a Linear Program (LP) using index notation. (Model)
- Implement a large scale LP in Gurobi. (Code)
- Make code runnable as a standalone command line tool. (Code)

1. Background

Nia is a data scientist at Trojan E-commerce, a medium sized online retailer with 17 fulfilment centers scattered across the US. This fiscal year, Trojan has the capital resources to build small-scale capacity expansions at a few of its fulfilment centers. However, as a company that prides itself in operational efficiency, Trojan wants to make sure to place the investment where it is needed the most. Nia has been assigned the task of identifying the top five fulfilment centers in which a small-scale capacity expansion would yield the greatest cost savings to Trojan's supply chain.

After much data cleaning and exploratory analysis, Nia decided to focus on minimizing the weekly outbound shipping cost from the fulfilment centers to customers. Trojan uses UPS 2-day delivery. Based on a regression analyses, she found that the cost of shipping one unit of item k from fulfillment center (FC) i to demand region j is $1.38 w_k \delta_{ij}$, where w_k is the shipping weight of the item (in lbs) and δ_{ij} is the distance from FC i to region j (in thousands of miles). The total weekly shipping cost for an item is this unit shipping cost multiplied by the number of units of item k shipped per week from FC i to region j. The objective to minimize is the sum of the above for all items.

While Trojan E-commerce sells hundreds of thousands of items, Nia conducted a clustering analysis to simplify the analysis and found 100 representative items, and she scaled up the demand for these so that they can serve as a proxy for all the items. Nia has also partitioned the US into 98 demand regions, and has estimated the weekly demand from each demand region for each of the representative items.

Trojan is committed to satisfying all customer demand for all of the items at all demand regions. The weekly demand for item k in region j is given as d_{jk}. However, Trojan can choose how much of this to provide from each FC. Using a closer FC would reduce the shipping cost, but capacity at each FC is limited. Since the company replenishes inventory every week, the amount of capacity required at a FC for fulfilling each unit of weekly demand of item k is equal to s_k, which is the storage size of item k (in cubit feet). The total capacity of FC i is given as q_i (in cubit feet).

1.1 Specific Question

To identify the top candidates for capacity expansion, Nia decided to conduct the following analysis:

1. Formulate a LP to minimize total transportation cost for all items, subject to not exceeding the total capacity at each FC and fulfilling the weekly demand for each item from every region. Assume that fractional number of units is allowed (so it's a LP, not a MIP).

2. Use the shadow price of capacity constraints to identify the top five FCs in which a small-scale capacity expansion would yield the greatest savings to weekly transportation cost.

2. Data

The following files are associated with this lab and can be downloaded from Blackboard.

- **data.xlsx**: the data Nia prepared for her analysis. The excel workbook has five worksheets, "Fulfillment Center", "Regions", "Distances", "Items", and "Demand." The following summarizes what is contained in each sheet.
  - Fulfilment Centers: the set of FCs, as well as capacity q_i for each FC i.
  - Regions: the set of demand regions.
  - Distances: the distance (in thousands of miles) δ_{ij} from each FC i to each region j. Each row represents a region and each column a FC.
  - Items: the set of items, as well as the shipping weight (in lbs) w_k and storage size (in cubit feet) s_k for each item k.
  - Demand: the demand d_{jk} at each region j for each item k. Each row represents an item and each column a region.
- **small_data.xlsx**: a toy dataset of the same format as the above, for development purposes.
- **output_for_small_data.xlsx**: the correct optimization output using the inputs from "small_data.xlsx".
- **visualizations.zip**: a compressed file containing visualizations Nia created for the three files above. These visualizations are not necessary for this lab but included to help you gain intuition and understand the data.
- **grade.py**: the grading script that will be used to automatically assess your "lab2_code.py" (see Section 3.2). You should run this before submitting your code to check what you will get (see instructions in Section 4).

3. Deliverables

Each individual must submit the following two files on Blackboard (individuals from the same team may submit the same files, but each individual is responsible for his/her own submission and will be graded separately.)

3.1 lab2_report.ipynb

A Jupyter notebook containing an executive summary and an abstract formulation of the linear program, correctly typesetted using Latex.

The **executive summary** must contain:

- The minimum total shipping cost for "data.xlsx" (the objective value of the LP). This must be correct to the nearest 1000 dollars.
- A ranking of the top five FCs with the most negative shadow prices. (Nia's recommendations of which FCs to invest in a small scale capacity increase so as to achieve the highest cost savings.) The first FC on this list should have the most negative shadow price, the second FC the second most negative, etc.

The **abstract formulation** must use the same notation for the data variables as given in Section 1, and define the data variables, the decision variables, the objective and the constraints. The formulation must use correct mathematical notation and be typeset using Latex in a Markdown cell. To express sums, the formulation should use the correct summation notation with Σ.

3.2 lab2_code.py

A Python module containing a **function called optimize** with two input parameters:

- **inputFile**: the path to the input file. (Examples of input files include "data.xlsx" and "small_data.xlsx".)
- **outputFile**: the path to the output file to be created by the function. (An example of an output file is "output_for_small_data.xlsx".

The output file created by the function must have the EXACT SAME FORMAT as the provided "output_for_small_data.xlsx". Precisely speaking, there must be three sheets, named "Summary", "Solution", and "Capacity Constraints" respectively.

The first sheet "Summary" should contain one column, with header "Objective Value", and a single entry which is the optimal cost.

The second sheet "Solution" should contain the columns "FC_name","region_ID","item_ID", and "shipment", describing how much of each item to ship per week from each FC to each region. You do not need to include rows in which shipment is zero.

The third sheet "Capacity Constraints" should contain the columns "FC_name" and "shadow_price", giving the shadow price of the capacity constraint for each FC.

The "lab2_code.py" must be able to be **run from command line** using the following command (in Anaconda prompt in Windows and in a Terminal in Mac):

```
python lab2_code.py inputFile outputFile
```

where "inputFile" should be replaced by the path to the input file, and "outputFile" should be replaced by the path to the output file. (For example, inputFile might be "small_data.xlsx" and outputFile might be "test_output.xlsx".)

4. Grading Rubric

The lab will be graded out of a total of 4 points, with one point in each of the following 4 categories. The grading is binary, meaning that a perfectly correct solution for a category would obtain 1 point, while any mistake (however minor) would result in 0 for that category.

i) Executive Summary

The executive summary in the Jupyter notebook clearly presents the minimum total shipping cost that is achievable (correct up to the nearest thousand dollars), as well as the top five FCs with the most negative shadow prices (correctly ordered so that the first FC on this list is the one with the highest cost savings, and the second FC the second highest, etc).

ii) Mathematical Formulation

The abstract formulation in the Jupyter notebook is correct, complete, and uses correct mathematical notation. All of the data variables must match the notation given in Section 1. The formulation must be formatted nicely in a Markdown cell using Latex.

For the formulation to be complete, you need to list all the data variables used, followed by the decision variables, the objective and the constraints. (See the following example from session 19, as well the Jupyter notebook version of the course handouts for the Latex code.)

Data:

- I: the set of books.
- J: the set of genres.
- a_{ij}: a binary variable denoting whether book i is of genre j. (These corresponds to the checkmarks in the original question.)
- q_j: how many books do we need of genre j.

Decision Variables: For each book $i \in I$, let x_i denote whether to carry the book. (Binary)
Objective and constraints:

$$\begin{aligned} \text{Minimize:} \quad & \sum_{i \in I} x_i \\ \text{subject to:} & \\ \text{(Enough books in genre)} \quad & \sum_{i \in I} a_{ij} x_i \geq q_j \quad \text{for each genre } j \in J. \end{aligned}$$

iii) Correct from Function Call

iv) Correct from Command Line

These are given by the output of the grading script `grade.py`. You should be able to run the following in a terminal (Anaconda Prompt in Windows or a Terminal in Mac) in the same directory as the data files and the `lab2_code.py` file.

```
python grade.py lab2_code.py
```

The output should be

```
...
Autograding Lab 2 script named "lab2_code.py":

    Correct from function call: 1
    Correct from command line: 1

Total code score (out of 2): 2
```

The grading script will point out where you have an error, you should repeatedly run the grading script until it is correct (or you will get whatever grade the script indicates).