ISyE 6669 Project - Team 7 - Fall 2020

Matthew Mendez, Connor Owen & Peter Williams

date: 2020-11-14

Introduction

You are in charge of developing an optimization solution for a large online retailer. The retailer sells K types of products. These products are stored in N warehouses spread around the country. Each warehouse has limited stock of products available. These are given in Warehouses.csv. We want to satisfy M customer orders. Each order may consist of varying quantities of different products. For now, you can assume that products are infinitely divisible.

For instance, all of the following are valid orders:

- *Order A requires 3 units of Product 1.*
- *Order B requires 5 units of Product 2 and 5 units of Product 3*
- *Order C requires 1.5 unit of Product 1*

Additionally, an order can be fulfilled from multiple warehouses. For example, if an order requires 2 units of Product 1 and 3 units of Product 2, you could send:

- *1 unit of Product 1 from Warehouse 1.*
- *1 unit of Product 1 and 1.5 units of Product 2 from Warehouse 2*
- *1.5 units of Product 2 from Warehouse 3*

The orders are given in Orders.csv.

Sending products from the warehouse to the customer has a cost. The cost is proportional to both the distance between the warehouse and the customer and the total weight of the items sent. The costs of sending one pound from a warehouse to satisfy a order are given in DeliveryCost.csv. The per-unit weight of the products is given in ProductWeight.csv.

Your goal is to assign customer orders to warehouses so that you satisfy all the orders while minimizing fulfillment costs.

Part A (50 pts)

Question 1 (20 pts)

Formulate a Linear Programming model to solve this problem. Your submission must be typed. Clearly define all variables, parameters, and constraints.

To begin we list products, warehouses, and orders according to the following notation:

$$k = 1, ..., K \text{ Products}, \tag{1}$$
$$j = 1, ..., N \text{ Warehouses, and} \tag{2}$$
$$i = 1, ..., H \text{ Orders} \tag{3}$$

We then define,

$$d_{ik} \in \mathbb{R} \text{ , the demand for product } k \text{ in order } i, \tag{4}$$
$$s_{jk} \in \mathbb{R} \text{ , stock of product } k \text{ in warehouse } j, \tag{5}$$
$$c_{ij} \in \mathbb{R} \text{ , cost of sending 1 lbs. of product from warehouse } j \text{ to customer } i, \tag{6}$$
$$w_k \in \mathbb{R}, (k = 1, ..., K) \text{ denotes the weight of product k in lbs.} \tag{7}$$

Our decision variables are formulated as,

$$x_{ijk} \in \mathbb{R}, \text{ number of units shipped of product } k \text{ from warehouse } j \text{ to customer } i, \text{ and} \tag{8}$$
$$\delta^*_{ik} = \text{ the number of units of product } k \text{ not fulfilled in order } i \tag{9}$$

and our objective is then to minimize the cost of shipping units to customers:

$$\min z = \sum_{i=1}^{H}\sum_{j=1}^{N}\sum_{k=1}^{K} x_{ijk}c_{ij}w_k + M\sum_{i=1}^{H}\sum_{k=1}^{K}\delta^*_{ik} \tag{10}$$

subject to,

$$\sum_{j=1}^{N} x_{ijk} + \delta^*_{ik} = d_{ik}, \forall i \in M, k \in K, \tag{11}$$
$$\sum_{i=1}^{H} x_{ijk} \leq s_{jk}, \forall j \in N, k \in K \tag{12}$$
$$x_{ijk} \geq 0 \forall i \in H, j \in N, k \in K \tag{13}$$
$$\delta^*_{ik} \geq 0 \ \forall i \in H, k \in K \tag{14}$$
$$M > 0, \text{ arbitrary large positive number (i.e. 'Big M').} \tag{15}$$

where (11) ensures that all demand must be satisfied or accounted for, (12) ensures that stock levels are not exceeded, and (13) and (14) ensure that non-negativity is enforced. =

Question 2 (20 pts)

Implement your model in Xpress or Gurobi/Python. Your program must read the data from the given files. Hard-coded data will be (severely) penalized. In your submission, this script should be named ModelA.mos or ModelA.py. Clearly explain your data structures and how different parts of your code correspond to different parts of your formulation. Specifically, explain which parts of the code generate the variables, objective function, and constraints. Your program's output should clearly show the optimal solution in an easy-to-read way.

We used *Gurobi/Python* to build our model, and our chosen data structures implemented in *ModelA.py* are derived from the imported dataset files. This allows our model to be run for different input files assuming that data input formats remain consistent. The python script relies on the following input data structures:

- The *orders.csv*, a file that is read into our environment as a *DataFrame* type object using the python *pandas* module. We denote this object as *orders_df* in our code. Based on the provided file, this object, in terms of rows and columns, is a 118×3 tabular object consisting of *Order ID*, *Product ID*, and *Quantity* containing information about demand for each product type. By aggregating the sum of *Quantity* along *Order ID* $(i = 1, ..., M)$ and *Product ID* $(k = 1, ..., K)$ we are able to formulate demand corresponding to our model where, $d_{ik} \in \mathbb{R}$, the demand for product k in order i.
- The *ProductWeight.csv* is read into our environment as a *DataFrame* type object denoted as *product_weight_df* in our code. In terms of rows and columns, *product_weight_df* is a 5×1 tabular object consisting of *Weight* for each product type. This allows us to implement the weight component of our formulation, that is, the data in this file provides $w_k \in \mathbb{R}, (k = 1, ..., K)$ the weight of product k in lbs.
- The *Warehouses.csv* is read into our environment as a *DataFrame* type object denoted as *warehouse_df* in our code. In terms of rows and columns, *warehouse_df* is a 40×3 tabular object consisting of *Warehouse ID*, *Product ID*, and *Stock*, i.e. the stock of each product in each warehouse. This allows us to implement the stock constraints in our formulation, that is, the data in this file provides $s_{jk} \in \mathbb{R}$, stock of product k in warehouse j.
- Lastly in our code, the *DeliveryCost.csv* is read into our environment as a *DataFrame* type object denoted as *costs_df* in our code. In terms of rows and columns, *costs_df* is a 8×46 tabular object consisting of *Warehouse ID* in the rows, and the associated cost of an *Order ID* in the columns. To simplify subsequent coding, we re-encode this data into a dictionary object denoted as *costs* which associates the cost of each order shipping from a warehouse. This data supports the formulation of $c_{ij} \in \mathbb{R}$, cost of sending 1 lbs. of product from warehouse j to customer i, since each *Order ID* corresponds to a customer.

With these input data objects, we create indices over which our code iterates over warehouses, orders and products to formulate demand, and stock constraints. However, we must deal with the complexity of not being able to fulfill all orders demanded based on stock levels. We solved for this using *Big M* notation above in our model, penalizing unmet demand, denoted in our model as $\delta^*_{ik} \geq 0 \ \forall i \in H, k \in K$. In our code, to account for this we add variables that account from the flow from a warehouse, or a product for an order. We then add constraints to our model that account for flow exceeding stock levels, and an overall constraint to accomodate total order δ. We set $M = 10,000$ in our script implementation to account for the total cost penalty, $M \sum_{i=1}^{H} \sum_{k=1}^{K} \delta^*_{ik}$.

Note that for the purposed of *Gurobi* implementation, all variables in this initial formulation as encoded as *GRB.CONTINUOUS*, or continuous type variables.

Question 3 (5 pts)

Solve your model. What is the objective function value of your solution? What does it mean in words? What is the optimal solution? Make sure to specify which orders are satisfied from which warehouse and what quantities of different items have been sent. In your write up, summarize your solution in a human readable format, e.g. a table.

Question 4 (5 pts)

Notice that all orders in the data require whole quantities of products. In practice, the products are not infinitely divisible. Suppose we were to impose this restriction (you do not have to implement it or change the formulation yet). Would the optimal solution change? If your answer is yes, would the optimal objective value be higher, lower, or stay the same? Why? If your answer is no, why not?

Part B (20 pts)

Question 5 (5 pts)

Now assume that products are not infinitely divisible and can only be sold in whole quantities. Fur- thermore, sending a package from a warehouse to a customer incurs a fixed cost in addition to the weight-based cost. Fixed costs are given in FixedCosts.csv. How would you change your formulation to reflect the new assumptions? You can either write out the new formulation or specify what you had to add and which parts of the old formulation had to be changed and how. Is your new formulation an LP or IP?

Question 6 (10 pts)

Implement the changes to your model in Question 5 in Xpress or Gurobi/Python. In your submission, this script should be named ModelB.mos or ModelB.py. As in Question 2, make sure you explain any new lines of code you have added.

Question 7 (5 pts)

Solve your model. What is the objective value of your solution? What does it mean in words? How does it compare to the solution of Question 3?

What is the optimal solution? Which orders are satisfied from which warehouse? What quantities of different items have been sent? In your write up, summarize your solution in a human readable format, e.g. a table.

Part C (30 pts)

Question 8 (15 pts)

Often businesses have to take into account considerations beyond just the cost. In our case, due to trade regulations, our warehouses and customer orders have been assigned to one of four different regions. These assignments are given in WarehouseRegions.csv and OrderRegions.csv. Orders from one region should be fulfilled by warehouses from the same region until the supplies are depleted. After that, they can be fulfilled from any region.

For instance, suppose Region 1 has only one warehouse, say Warehouse 1. This warehouse holds 5 units of Product 1. Also suppose there is only one order coming from Region 1, say Order 1. Order 1 requires 7 units of Product 1. With the new regional constraints, we have to send 5 units of Product 1 from Warehouse 1 to Order 1. We can satisfy the remaining demand of 2 units form any warehouse. Notice that this constraint takes precedence over any cost considerations, i.e. we have to send 5 units from Warehouse 1 even if it is cheaper to satisfy the order from another warehouse.

Write the model for the updated problem. Explain any additional parameters, variables, and constraints you had to introduce. Your new formulation should include the changes you have made in Question 5. You don't have to explain these again.

Question 9 (10 pts)

Implement your new model in Xpress or Gurobi/Python. In your submission, this script should be named ModelC.mos or ModelC.py. As in Question 2, make sure you explain any new lines of code you have added.

Question 10 (5 pts)

Solve your model. What is the objective value of your solution? What does it mean in words? How does it compare to the solution of Question 8? What is the optimal solution? Which orders are satisfied from which warehouse? What quantities of different items have been sent? In your write up, summarize your solution in a human readable format, e.g. a table.