ISyE 6669 Project - Team 7 - Fall 2020

Connor Owen, Matthew Mendez & Peter Williams

date: 2020-11-16

Contents

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,

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

Our decision variables are formulated as,

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

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 have been encoded as 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.

Based on our implementation in *Gurobi* in *ModelA.py* as described above our realized objective function value for minimization is $z = 165497.8112$ *(rounded)*. However, because we used a *'Big M'* model formulation to account for 16.0 unfulfilled orders with $M = 10,000$ our total order cost is in fact, $165497.8112 - 160000 = 5497.8112$

In terms of our decision variables and output, our model provides two key outputs, the quantity of products planned for fulfillment from each warehouse, for each product *(Table 1)*. It also provides the unfulfilled demand quantity for each product and order *(Table 2)*.

The following provides a preview of both of these tables with actual results included,

Table 1: Preview: Minimizing Order Fulfillment Costs

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
1	3	2	2.0
1	4	2	3.0
.	.	.	.
.	.	.	.
45	8	3	1.0
45	8	4	1.0

Note that the full result for part *A* can be found in the appendix for reference.

Our model output also provides a summary, as mentioned of unfulfilled orders which is provided in total here:

Table 2: Preview: Unfulfilled Orders Results

Order ID	Product ID	Quantity Unfulfilled
17	5	1.0
20	1	3.0
20	3	2.0
31	1	2.0
31	3	3.0
31	5	3.0
39	5	1.0
43	5	1.0

Note that based on the results from our model, the total unfulfilled order demand quantity is 16.0 products.

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?

Normally when we are working on a minimization problem imposing the restriction of product quantity always taking integer values, our optimal objective could be higher or stay the same. However, the general formulation of our problem above is in the form of a transportation problem. That is because we have:

- A supply or stock of product k in warehouse j yielding our supply points,
- and have i demand points for each customer order.
- We also have a variable cost to ship product k from warehouse j to customer i, and our key decision variable is,
- $x_{ijk},$ number of units shipped of product k from warehouse j to customer i.

As a result, imposing this restriction does not change our solution in our problem.

Part B (20 pts)

Question 5 (5 pts)

Now assume that products are not infinitely divisible and can only be sold in whole quantities. Furthermore, 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?

The inclusion of a fixed cost component to our model would require the addition of a variable resulting in two cost variables, the previously specified cost,

$$c_{ij} \text{ , cost of sending 1 lbs. of product from warehouse } j \text{ to customer } i, \tag{16}$$

and the new fixed cost variable for each warehouse, customer combinations that we would specify as,

$$f_{ij} \text{ , the fixed cost of sending 1 lbs. of product from warehouse } j \text{ to customer } i. \tag{17}$$

As a result of this variable addition we would also need to re-formulate our objective function including this new cost as:

$$\min z = \sum_{i=1}^{H}\sum_{j=1}^{N}\sum_{k=1}^{K} x_{ijk}(c_{ij} + f_{ij})w_k + M\sum_{i=1}^{H}\sum_{k=1}^{K}\delta_{ik}^{*}. \tag{18}$$

This new objective would also be subject to the our key decision variable taking on whole number or integer values as specified below:

$$x_{ijk} \in \mathbb{N}, \text{ number of units shipped of product } k \text{ from warehouse } j \text{ to customer } i. \tag{19}$$

We can also add notation to our demand and stock variables to explicitly specify they take on whole number values (integer) as follows:

$$d_{ik} \in \mathbb{N} \text{ , the demand for product } k \text{ in order } i, \tag{20}$$
$$s_{jk} \in \mathbb{N} \text{ , stock of product } k \text{ in warehouse } j, \tag{21}$$
$$\tag{22}$$

As described above, this problem is formulated as a transportation problem, which can be considered here as an integer programming problem (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.

In order to solve our problem given changes specified in problem 5, we add some key changes to our code including:

- Reading in a *DataFrame* object from the *FixedCosts.csv* file provided via the command, *fixed_ costs_ df = pd.read_ csv('../csv/FixedCosts.csv')* that provides a fixed cost for each combination specified of order and warehouse, i.e. f_{ij} , the fixed cost of sending 1 lbs. of product from warehouse j to customer i. This data presents itself in our model as a variable in *Gurobi* as *fixed_ cost_ ind = m.addVars([(w,o) for w in warehouses for o in orders], vtype=GRB.BINARY)* through flow indices that specifies the cost inclusion (binary) when an order is paired with a warehouse in our decision variable.

- Ensuring that our key decision variable, namely, x_{ijk}, is constrained to take on integer values only via the *Gurobi* vtype argument, $flow = m.addVars(indices, lb = 0, vtype = GRB.INTEGER)$.

- Including an indicator for maximum stock for specific product so as to ensure that fixed costs are included for a warehouse in consideration given our order. This is included with the *max_ stock = max(warehouse_ stock[w][k] for w in warehouses for k in products)* list in our variables. This fixed costs is activated for a warehouse in the order via the following constraint: *m.addConstr(fixed_ cost_ ind[(w,o)] >= sum(flow[(w, o, k)] for k in products if k in orders_ data[o])/max_ stock)* in line 82.

We also enhanced our ModelB.py script to include unit costs in the output for each order, and in addition provide leftover supply for additional insights.

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?

With the inclusion of fixed costs and new variable type designations our model provide the following solution. In this topline solution we provide the realized value from our objective function, as well as the total order costs, which are the sum of total fixed costs + unit costs. In this case it is work noting that our objective function value can also be understood as the product of *Unfulfilled Demand Quantity* $\times$ M + *Total Order Cost*:

Table 3: Model B: Topline Results (Rounded)

Model Result	Value
Objective Function Value	169,973.95
Total Order Cost	9,973.95
Total Fixed Cost	4,215.00
Total Unit Costs	5,758.95
Penalty	M = 10,000
Unfulfilled Demand Quantity	16

It is worth noting that our solution yields the same level of unfulfilled orders, with an order quantity of 16. Our total costs have increased with the inclusion of fixed costs of $4,215.00$, and units costs $5,758.95$ slightly higher than the unit costs in problem 3 5497.81. As a result our overall objective function value is higher than in problem 3, with a value of $169,973.95$ vs. the previously achieved 165497.81

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.

The full output of our newly specified model includes Order ID, Product ID and Warehouse ID quantity assignments in full. Given the size of the table, a preview is provided here. The full table can be found in the appendix:

Table 4: Preview Solution, Problem B: Minimizing Order Fulfillment Costs

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
1	3	2	5
2	4	3	4
..	..	..	..
..	..	..	..
20	7	3	2
..	..	..	..
..	..	..	..

Based on the full solution output, we can aggregate solution quantity fulfilled across each Warehouse ID to determine total quantity for each location. This data is summarized below, and a list of distinct Order IDs fulfilled from each warehouse is also provided in full:

Table 5: Model B: Warehouse Fulfillment

Warehouse ID	Orders IDs (Partially or Fully Fulfilled)	Total Quantity Fulfilled
1	5,9,31,33,39,44	30
2	10,27,28,30,42,43	35
3	1,3,4,17,18,35,37	33
4	2,7,12,14,16,22,41	37
5	13,17,24,32,36,38,44	39
6	7,14,17,19,25,34,40,43	36
7	8,11,15,20,24,26,41	34
8	6,21,22,23,29,33,36,45	37
	Total Quantity	281

The full solution output can also be aggregated across each Product ID to determine total quantity for each product type or item. This data is summarized below with the sum of quantity fulfilled for each product type:

Table 6: Model B: Product Fulfillment

Product ID	Quantity Fulfilled
1	54
2	57
3	62
4	48
5	60
Total Quantity	281

Note that the total unfulfilled order quantity with associated Product ID is also provided in the appendix for review.

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.

Appendix

Complete solution, problem A including warehouse, product and order assignments and quantity:

Table 7: Full Solution Problem A: Minimizing Order Fulfillment Costs (1/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
1	3	2	2
1	4	2	3
2	4	3	4
2	4	4	1
3	2	5	2
3	3	2	3
3	3	3	2
4	2	1	1
4	3	1	1
4	8	3	2
5	1	3	1
5	1	5	2
6	8	5	3
7	4	2	2
7	6	2	3
8	5	5	1
8	7	2	1
9	1	1	4
9	1	3	1
9	1	5	2
10	1	3	3
10	2	2	1
10	2	3	5
10	2	4	1
11	7	2	4
12	4	1	3
12	4	5	3
13	5	5	4
14	4	4	2
14	5	1	2
14	5	4	2
14	6	1	1
15	6	3	1
15	7	5	3
16	3	2	1
16	4	4	4
17	3	5	5
17	5	5	1
17	6	1	1
17	6	3	4
17	6	4	3
18	3	1	3
18	3	5	1
19	6	4	3
20	7	1	1
20	7	3	1

Table 8: Full Solution Problem A: Minimizing Order Fulfillment Costs (2/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
21	8	1	2
21	8	4	1
22	4	3	5
22	8	3	1
23	8	1	4
23	8	3	1
23	8	5	4
24	5	1	2
24	5	2	2
24	7	2	3
25	6	2	2
25	6	3	3
25	6	5	2
26	7	1	3
26	7	4	5
26	7	5	3
27	2	1	2
27	2	5	2
27	4	1	2
27	4	5	1
28	2	3	5
29	8	4	4
29	8	5	1
30	2	1	3
30	2	2	4
31	1	1	1
32	3	1	2
32	3	3	2
32	5	3	1
33	1	4	6
33	8	3	3
34	5	5	1
34	6	2	3
34	6	3	1
34	6	5	3
35	3	1	3
36	5	1	5
36	5	2	4
36	5	4	4
37	3	3	2
37	3	5	4
38	2	4	4
38	5	2	4
38	5	4	1
39	1	2	3
39	1	5	3
40	6	1	3
40	8	2	2

Table 9: Full Solution Problem A: Minimizing Order Fulfillment Costs (3/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
41	4	1	3
41	4	3	1
41	4	5	1
41	7	1	1
41	7	3	6
41	7	5	3
42	4	4	2
43	2	4	4
43	2	5	3
43	5	3	1
43	6	2	2
43	6	3	1
44	1	2	2
44	1	5	2
44	3	2	2
44	5	3	4
45	8	1	1
45	8	2	4
45	8	3	1
45	8	4	1

Table 10: Problem A: Unfulfilled Orders Results (1/1)

Order ID	Product ID	Quantity Unfulfilled
17	5	1.0
20	1	3.0
20	3	2.0
31	1	2.0
31	3	3.0
31	5	3.0
39	5	1.0
43	5	1.0

Complete solution, problem B including warehouse, product and order assignments and quantity based on fixed cost inclusion model:

Table 11: Full Solution Problem B: Minimizing Order Fulfillment Costs (1/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
1	3	2	5
2	4	3	4
2	4	4	1
3	3	2	3
3	3	3	2
3	3	5	2
4	3	1	2
4	3	3	2
5	1	3	1
5	1	5	2
6	8	5	3
7	4	2	4
7	6	2	1
8	7	2	1
8	7	5	1
9	1	1	4
9	1	3	1
9	1	5	2
10	2	2	1
10	2	3	8
10	2	4	1
11	7	2	4
12	4	1	3
12	4	5	3
13	5	5	4
14	4	1	1
14	4	4	4
14	6	1	2
15	7	3	1
15	7	5	3
16	4	2	1
16	4	4	4
17	3	1	1
17	3	5	3
17	5	5	3
17	6	3	4
17	6	4	3
18	3	1	3
18	3	5	1
19	6	4	3
20	7	1	3
20	7	3	2

Table 12: Full Solution Problem B: Minimizing Order Fulfillment Costs (2/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
21	8	1	2
21	8	4	1
22	4	3	3
22	8	3	3
23	8	1	4
23	8	3	1
23	8	5	4
24	5	1	2
24	5	2	2
24	7	2	3
25	6	2	2
25	6	3	3
25	6	5	1
26	7	1	2
26	7	4	5
26	7	5	3
27	2	1	3
27	2	5	3
28	2	3	2
29	8	4	4
29	8	5	1
30	2	1	3
30	2	2	4
31	1	1	1
31	1	3	2
32	5	1	2
32	5	3	3
33	1	4	6
33	8	3	3
34	6	2	3
34	6	3	1
34	6	5	4
35	3	1	3
36	5	1	5
36	5	2	2
36	5	4	2
36	8	2	2
36	8	4	2
37	3	3	2
37	3	5	4
38	5	2	4
38	5	4	5
39	1	2	3
39	1	5	3

Table 13: Full Solution Problem B: Minimizing Order Fulfillment Costs (2/3)

Order ID	Warehouse ID	Product ID	Quantity Fulfilled
40	6	1	3
40	6	2	2
41	4	1	4
41	4	3	3
41	4	5	2
41	7	3	4
41	7	5	2
42	2	4	2
43	2	4	4
43	2	5	4
43	6	2	2
43	6	3	2
44	1	2	2
44	1	3	1
44	1	5	2
44	5	2	2
44	5	3	3
45	8	1	1
45	8	2	4
45	8	3	1
45	8	4	1

Table 14: Problem B: Unfulfilled Orders Results (1/1)

Order ID	Product ID	Quantity Unfulfilled
17	5	1.0
20	1	1.0
20	3	1.0
25	5	1.0
26	1	1.0
27	1	1.0
28	3	3.0
31	1	2.0
31	3	1.0
31	5	3.0
39	5	1.0