ISyE 6669 Project - Team 7 - Part B - Fall 2020

Connor Owen, Matthew Mendez & Peter Williams

date: 2020-11-18

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,

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

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

$$f_{ij} \text{ , the fixed cost of sending 1 lbs. of product from warehouse } j \text{ to customer } i. \tag{2}$$

We also need to add a binary indicator variable, to ensure we have a decision variable that accounts for a warehouse being chosen to fulfill and order, namely,

$$y_{ij} = 1 \text{ , if warehouse } j \text{ sends an order to to customer } i, y_{ij} = 0 \text{ otherwise.} \tag{3}$$

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

$$\min z = \sum_{i=1}^{H}\sum_{j=1}^{N}\sum_{k=1}^{K} x_{ijk}c_{ij}w_k + \sum_{i=1}^{H}\sum_{j=1}^{N} f_{ij}y_{ij} + M\sum_{i=1}^{H}\sum_{k=1}^{K}\delta_{ik}^{*}. \tag{4}$$

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

$$x_{ijk} \in \mathbb{N}, \text{ number of units shipped of product } k \text{ from warehouse } j \text{ to customer } i. \tag{5}$$

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

$$d_{ik} \in \mathbb{N} \text{ , the demand for product } k \text{ in order } i, \tag{6}$$
$$s_{jk} \in \mathbb{N} \text{ , stock of product } k \text{ in warehouse } j, \tag{7}$$
$$\tag{8}$$

As described above, this problem is formulated like 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, $flow = m.addVars(indices, lb = 0, vtype = GRB.INTEGER)$. It is also worth noting that the incoming demand is assumed to always take on integer values.

- 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 in line 53. 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.

To simplify readibility of the code we also added the following

Code that generates constraints: # Define cost variable m.addConstr(cost == sum(flow[t] *costs[t[0]][t[1]]*product_weights[t[2]] for t in indices)) # meet demand for o in orders: for k in orders_data[o]: m.addConstr(sum(flow[(w, o, k)] for w in warehouses) + order_delta[(o, k)] == orders_data[o][k]) # don't exceed stock for w in warehouses: for k in products: m.addConstr(sum(flow[(w, o, k)] for o in orders if k in orders_data[o]) + supply_delta[(w,k)] == warehouse_stock[w][k]) # define order delta total m.addConstr(tot_unfulfilled_demand == sum(order_delta[t] for t in order_delta_indices)) # define leftover supply total m.addConstr(tot_leftover_supply == sum(supply_delta[t] for t in supply_delta_indices))

Code that generates objective function: m.setObjective(cost + 10000 * (tot_unfulfilled_demand+tot_leftover_supply), sense=GRB.MINIMIZE)

Code that generates variables; order_delta = m.addVars(order_delta_indices, lb=0, vtype=GRB.CONTINUOUS) supply_delta = m.addVars(supply_delta_indices, lb=0, vtype=GRB.CONTINUOUS) tot_unfulfilled_demand = m.addVar(lb=0, vtype=GRB.CONTINUOUS) # this is the extra variable that I mentioned above tot_leftover_supply = m.addVar(lb=0, vtype=GRB.CONTINUOUS)

Code that generates constraints: # Define cost variable m.addConstr(cost == sum(flow[t]*costs[t[0]][t[1]]*product_weights[t[2]] for t in indices)) # meet demand for o in orders: for k in orders_data[o]: m.addConstr(sum(flow[(w, o, k)] for w in warehouses) + order_delta[(o, k)] == orders_data[o][k]) # don't exceed stock for w in warehouses: for k in products: m.addConstr(sum(flow[(w, o, k)] for o in orders if k in orders_data[o]) + supply_delta[(w,k)] == warehouse_stock[w][k]) # define order delta total m.addConstr(tot_unfulfilled_demand == sum(order_delta[t] for t in order_delta_indices)) # define leftover supply total m.addConstr(tot_leftover_supply == sum(supply_delta[t] for t in supply_delta_indices))

Code that generates objective function: m.setObjective(cost + 10000 * (tot_unfulfilled_demand+tot_leftover_supply), sense=GRB.MINIMIZE)

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 1: 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 2: 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 3: 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 4: 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.

Appendix

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

Table 5: 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 6: 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 7: 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 8: 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