ISyE 3133 Team Project

PART A

Alejandro L. Hernandez, Jaclyn N. Nichols, Esteban Ulloa

Introduction

We are asked to minimize fulfillment costs by optimizing how we assign a set of orders that need certain products to a set of warehouses that contain a specific amount of inventory for each product. In this project, we create a linear program to solve the problem, code the linear program, and discuss the relation of a linear program to an integer program.

We will create our linear program by:

1. Deriving variables
2. Listing data
3. Creating an objective function (minimization) using variables
4. Defining constraints

Solution

1. **Variables**
   - $X_{K,N,M}$ = # of units of products of type K fulfilled by the warehouse N for order M, $\forall_K \forall_N \forall_M$
2. **Data**:
   - M = # of orders
   - N = # of warehouses
   - K = # of product
   - D_{NM} = $ cost/lb to deliver order M from warehouse N, $\forall_N \forall_M$
   - O_{MK} = # of product K in order M, $\forall_M, \forall_K$
   - W_K = weight of product K, $\forall_K$
   - S_{NK} = # of product K stock in warehouse N $\forall_N \forall_K$
3. **Objective Function**:

   Let M = 1...t, N = 1...u, and K = 1...v where t,u,v $\in \mathbb{N}$, then

   $\min z = \Sigma_M\Sigma_N\Sigma_K X_{K,N,M}\ W_k D_{N,M}$
4. **Constraints**:

   S.T.
   - $\Sigma_N X_{K,N,M} = O_{M,K}$ [Each order fulfilled once constraint]
   - $\Sigma_M X_{K,N,M} \leq S_{N,K}$ [warehouse stock constraint]
   - $\Sigma_M\Sigma_N\Sigma_K X_{K,N,M} \geq 0$ [nonnegativity constraint]

We have sent our code, Appendix A, separately and attached a copy at the end. We have also attached a copy of the excel spreadsheet, Appendix B, of our solution to the end as well.

Observations

- What is the objective value of your solution?
    - The objective value of our solution is 8.862828759e+03
- What does it mean in words?
    - This means that 8.862828759e+03 is the minimum cost of fulfilling all the products in all the orders through the optimal warehouse.
- What is the optimal solution, i.e. which orders are satisfied from which warehouse?
    - Please refer to Appendix B for a full list of which orders are satisfied from which warehouses. However, we have provided a short overview below on how many orders are fulfilled at each warehouse.

Warehouse #	# of Orders Satisfied
1	34
2	28
3	26
4	27
5	27

It takes 142 partitions of our set of orders to minimize our delivery costs. A majority of the orders are satisfied by Warehouse 1.

- What quantities of different items have been sent?
    - Please refer to Appendix B for exact pairings; however, the table below shows a small snapshot of the quantities of the different items sent.

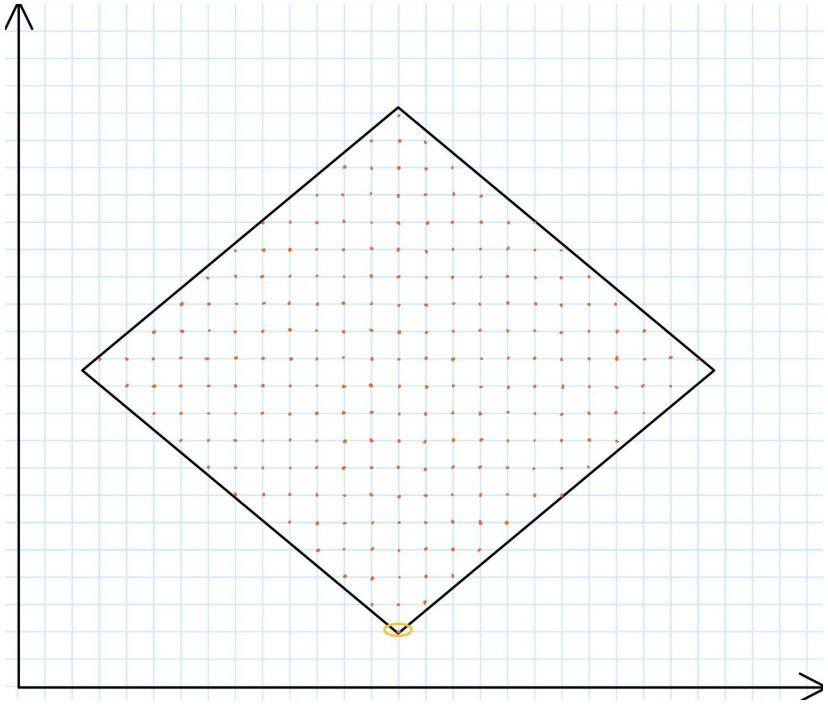
Item #	# of items shipped
1	50
2	34
3	35
4	23

It is clear that item 1 was fulfilled proportionally more than orders 2 through 4. Approximately, 35 % of the items shipped were item 1.

LP/IP Comparison

No, the optimal solution will not change because we are not given any non-integers in our problem set. Since we are only given whole numbers, and our optimal solution has only whole numbers, we will be able to use the same amount of inventory in each warehouse to optimize our solution as if we were using integer optimization.

Below we have included two examples to help better explain our point. Example one is a visual of the minimization between a linear program and the points of an integer program.



As you can see, the optimal point is at the bottom of our linear program and is also the solution to the integer program. Like in the project model, the optimal solution for the linear and integer program is the same.

The following example shows why the optimization would change only if we were given fractional parts of products for our linear program versus an integer program.

Example:

Warehouse 1

Product	Stock
1	5
2	7
3	9

Warehouse 2

Product	Stock
1	8
2	2
3	4

We need to fulfill the following orders.

Order 1:

- Product 1: 3.5
- Product 2: 2
- Product 3: 8

Order 2

- Product 1: 4
- Product 2: 3
- Product 3: 1

If we take partials then we could take all of the products 2 and 3 from Warehouse 1 and only 2.5 of product 1 from Warehouse 2; however, if we are not allowed partials, this number changes. Since we cannot take part of an item, then presumably we will have to take 4 units of product 1 to fulfill the order and thus meaning will now take 3 items from Warehouse 2 instead of 2.5 products.

Appendix

Appendix A

```python
from gurobipy import GRB, Model, quicksum
from csv import reader, writer
import pandas as pd
#DATA:
product_list = set()
order_list = set()
warehouse_list = set()
'''Creates dictionary that maps each product to its weight'''
p = {}
with open("ProductWeight.csv") as win:
    weightreader = reader(win)
    next(weightreader)
    for product in weightreader:
        product_list.add(int(product[0]))
        p[int(product[0])] = int(product[1])
'''
    Creates o:
    Dictionary mapping (order, product) to the quantity of each product in an
    order.
'''
orders_df = pd.read_csv("Orders.csv", header = 0)
o = {}
for index, row in orders_df.iterrows():
    order_list.add(int(row[0]))
    o[row[0], row[1]] = row[2]
'''
    Creates d:
    Dictionary mapping (warehouse, order, product) to the price of fulfilling
    that product in the order for each warehouse.
'''
delivery_cost = pd.read_csv('DeliveryCost.csv', header = 0, index_col='Warehouse ID/OrderID')
d = {}
for warehouse_ID, row in delivery_cost.iterrows():
    warehouse_list.add(int(warehouse_ID))
    for order in range(len(row)):
        for product in product_list:
            d[warehouse_ID, order + 1, product] = p[product] * row[order]
'''
    Creates s:
    Dictionary mapping (warehouse, product) to the stocked quantity of a
    product that each warehouse has
'''
warehouses = pd.read_csv('Warehouses.csv', header = 0)
s = {}
for index, row in warehouses.iterrows():
    s[row[0], row[1]] = row[2]
#MODEL:
model = Model("Retail Model")
#VARIABLES:
x = model.addVars(product_list, order_list, warehouse_list, name = 'x')

#OBJECTIVE:
model.setObjective(quicksum(x[k, m, n] * d[n, m, k] for n, m, k in d.keys()), sense = GRB.MINIMIZE)

#CONSTRAINTS:
model.addConstrs(quicksum(x[k, m, n] for n in warehouse_list) == o[m, k] for m, k in o.keys())
model.addConstrs(quicksum(x[k, m, n] for m in order_list) <= s[n, k] for n in warehouse_list for k in product_list)

model.optimize()
```

Appendix B

Warehouse	Order	Product	Product Quantity
1	2	1	1
1	8	8	3
1	10	1	4
1	12	4	2
1	13	7	1
1	16	6	2
1	17	2	2
1	18	7	2
1	20	5	2
1	21	6	2
1	22	6	3
1	23	3	1
1	24	3	1
1	24	8	1
1	26	2	2
1	27	5	4
1	29	10	3
1	30	1	1
1	30	9	1
1	31	9	4
1	32	3	4
1	33	8	1

1	34	1	1
1	34	10	3
1	36	6	3
1	37	7	2
1	40	5	1
1	41	1	1
1	41	7	2
1	43	2	3
1	44	4	4
1	45	4	1
1	45	8	1
1	47	7	2
2	4	5	4
2	5	9	4
2	7	7	3
2	9	4	2
2	9	8	2
2	10	1	2
2	14	5	1
2	16	4	1
2	17	3	2
2	19	3	3
2	21	1	3
2	23	3	3

2	25	8	3
2	26	2	4
2	29	10	3
2	35	1	1
2	36	6	3
2	37	5	1
2	38	9	1
2	41	10	1
2	43	2	4
2	44	4	3
2	45	8	4
2	46	7	4
2	47	5	1
2	47	7	3
2	48	6	3
2	50	5	1
3	1	5	4
3	2	1	3
3	3	2	3
3	4	5	2
3	5	9	1
3	6	10	3
3	7	7	3
3	9	4	3

3	10	1	1
3	11	3	2
3	14	3	2
3	15	5	2
3	15	7	2
3	17	2	4
3	25	8	3
3	28	2	3
3	28	8	1
3	30	1	2
3	31	9	1
3	33	4	3
3	34	10	3
3	36	6	4
3	38	9	1
3	41	10	1
3	47	3	3
3	49	8	1
4	1	7	2
4	2	1	1
4	3	2	4
4	6	7	1
4	8	2	1
4	8	8	2

4	9	5	2
4	9	6	1
4	17	5	1
4	23	5	3
4	24	3	4
4	26	2	2
4	28	2	3
4	30	1	4
4	32	3	1
4	33	3	1
4	33	8	3
4	34	10	1
4	38	9	2
4	39	4	3
4	41	10	4
4	43	5	1
4	44	6	4
4	45	4	3
4	45	6	1
4	48	6	2
4	49	8	1
5	3	2	2
5	10	1	2
5	11	2	2

5	12	4	2
5	15	7	4
5	17	2	1
5	18	10	1
5	20	6	1
5	22	6	3
5	23	3	1
5	24	3	2
5	25	8	3
5	26	2	1
5	27	5	1
5	29	10	4
5	31	9	4
5	35	6	1
5	36	6	1
5	39	4	1
5	40	3	3
5	41	1	2
5	42	5	2
5	42	7	1
5	43	2	2
5	44	4	4
5	45	4	1
5	50	5	3