Supply Chain Optimization in Pyomo/Gurobi

Aug 11, 2017

Dale Drasnin	20529695
Ahilan Jeyaraman	20432293
Vincent Magas	20469866
Stefanno Polo	20508389

1 - Introduction

The purpose of this report is to convey and present the methodology and approach that has been taken in building a supply chain for the provided case. The overall goal of the project has been to analyze the existing resources available to ABC Company and to build a supply chain that will minimize the total cost of the operation. There are two manufactured products, A and B which are distributed across a potential 32 customers. These two products are manufactured in 12 potential plants which are then supplied by 22 potential vendors. This project creates the supply chain from the bottom-up.

Within the scope of this project is the proper management and option selection of the following key elements: Customer groups, price planning, supplier or vendor sourcing, facility production, and transportation across the supplier-to-facility, and facility-to-customer. The first and foremost of this, the customer groups, deal with varying demands across three different price points for both products. As demand is directly dependent on the unit price of each product, this is the first item investigated within the project. Following this, the total number of facilities required to meet the total demand, as well as which facility locations should be opened is investigated. As both of these products contain multiple subassemblies that contain parts supplied by external vendors, the total number of vendors required are also determined. Finally, cost-effective routes between vendors-to-facilities, and facilities-to-customers are created and evaluated, completing the initial solution for the supply chain.

2 - Project Information & Assumptions

2.1 - Products

There are two products that are being considered in this project: Product A, and Product B. These products have been provided with their respective product trees featured in Figure 1 and 2 below. Due to the main assembly containing a one to one ratio with all of its subcomponents, the team considers that Product A requires 8 unique subcomponents per unit (A111, A112, A121, A13, A211, A212, A22, A23) and Product B requires 7 unique subcomponents per unit (B11, B12, B13, B211, B212, B22, B3). The team assumes that each of these unique subcomponents can be ordered from the vendors based on their availability.

2.2 - Sourcing

In sourcing the various unique subcomponents for Products A and B, there are 22 potential suppliers the can be utilized. As stated in the provided information, the cost of sourcing each unit from the supplier includes the cost of both price and shipping, as such it is assumed that any transportation costs due to distance is already included in the provided cost. Each of these suppliers also have their own capacities, given in thousands for each subcomponent that they can provide.

2.3 - Customers

There are 32 potential customers in total that is considered within the supply chain. Each customer has a unique location indicated by x & y coordinates, which determines their distance from the nearest facilities to serve them. The distance between each customer and plant is used in calculating their transportation costs, these distances are calculated as Euclidean distances. Each of the 32 customers also have varying demands for both Product A and B depending on the unit price that they are sold at. In the implementation of the model in python, the customers are designated the variable *j*.

2.4 - Plants

In this case, there are a total of 13 potential plant locations. These locations are given based on their x and y coordinates in a Cartesian Grid as well as their unique weekly operating costs. The operating costs are denoted as R_i in the python implementation of the model. As previously mentioned regarding customer locations, the distance between these plants and their customer locations are calculated as Euclidean distances. In the implementation of the model in python, the plants are designated the variable *i*.

2.5 - Transportation

For the aspect of transportation in the project, trucks with fixed capacities of 50,000 units for Product A or 30 000 units for Product B with the possibility of mixed shipments are considered, these parameters are denoted as TC_k in the python implementation of the model.

Each new truck that is required for a run is given a fixed cost of $25000, while the variable cost for each product is calculated based on the provided formula given in Equation 1 below. In this equation, N_A and N_B refer to the number of units of Product A and Product B are carried in a given truck, while T_A = $0.0015 and T_B = $0.0020 are the actual costs per kilometer travelled for Product A and Product B. In the implementation of the model in python, the distance travelled of a truck is denoted as D[(i,j)] (Distance from Plant *i* to Customer *j*).

Figure 1 - Variable Cost of a Single Truck for Each Run

$$\frac{[(NA * TA) + (NB * TB)] * D[(i,j)]}{2}$$

3 - Problem Description

The goal of this project is to develop a supply chain that addresses all of the requirements and constraints discussed in Section 2 while maximizing the total profit of the supply chain. This must be achieved by employing methodologies learned in MSCI 434 to understand the problem and develop a solution approach for all parts of the project breakdown then coding those solution methodologies into a Python model to solve the system. An important assumption made during the creation of this model is that demand is not required to be met completely if that results in a less optimal solution. Adhering to this assumption changes the solution drastically and allows for greater flexibility of the solution and therefore an increased likelihood of producing a higher profit, or a more realistic and optimal solution.

The end solution accounts for the selection of active manufacturing plants, the assignment of production quantities to manufacturing plants, the transportation of items from the manufacturing plants to the customers to meet demand, the selection of active raw material suppliers, and the assignment of order quantities for each active material supplier. The total profit of the system accounts for the profit from satisfying demand, the cost of production, the cost of transportation, and the cost of materials from suppliers.

The final solution must output the total profit of the system, the active manufacturing plants, production quantities for each active manufacturing plant, transportation information for each manufacturer-customer item flow, active suppliers, and the quantities ordered from each supplier.

4 - Solution Approach

4.1 - Problem Breakdown & Approach

This problem is broken into two main sections that must be solved in order. The first section of the problem can be designed as a hub allocation problem. This portion of the project provides the number of plants that will be required, the number of products produced per plant, and subsequently the flow of goods from plant i to customer j. Additionally, the required number of trucks for each plant-to-customer pair is determined in this portion of the problem. The second section of the breakdown deals with the other half of the supply chain, which is the supplier to plants matching. This second section takes in as input the production demands of each of the plants for Product A and Product B from the previous section, as well as and the max capacity parameters for each of the subcomponents for each supplier and the associated pricing. This second section then outputs each of the Supplier-to-Plant flow for both Products A and B by minimizing purchase costs.

There are total of 9 pricing combination between products A and B (e.g. Product A: $4.6 & Product B: $5.8) and their resulting demand is run through the python model. Each pair of pricing choices results in different demands and revenue and therefore different costs. The pricing combination which results in the greatest profit is then chosen as the solution.

4.1.1 - Hub Allocation (Manufacturers & Customers)

The first part of the problem is deciding how much of the demand to fulfill and with which manufacturing plants. This step focuses on the generation of an MIP model that will allocate specific customers to factories. Not all customers will be selected, not all demands will be meet and not all manufacturers will be utilized. The objective function of this model is to maximize the profits given number of items supplied to customers and the price to supply those customers based on distance from manufacturer.

Sets:

- I: Manufacturers
- J: Customers
- K: Products

Parameters:

- MCi: Running costs for manufacturer i
- Rk: Manufacturing hours for 1000 units of k
- Dij: Distance from manufacturer i to customer j
- Tk: Transport cost of product k
- CDjk: Demand of product k for customer j
- Pk: Selling price of k
- TCk: Truck capacity for k
- ULk: Supplier upper limit for product k
- MCi: Cost of running manufacturer k

Variables:

- Xijk: Products k supplied from manufacturer i to customer j
- Mi: Binary variable representing existence of manufacturer i
- NTij: Number of trucks required from i to j (Rounded Up)
- ENTij: Exact number of trucks required from i to j

Constraints:

Figure 2 - Hub Allocation Constraints

$$\sum_{i=0}^{12} Xijk \;<=\; CDjk;\; for\ all\ j\ and\ k$$

$$\sum_{i=0}^{12}\sum_{j=0}^{31} Xijk \;<=\; ULk\ for\ all\ k$$

$$\sum_{j=0}^{31}\sum_{k}^{A,B} Xijk * (Rk/1000) \;<=\; 25000 * Mi\ for\ all\ i$$

$$NTij <= (\tfrac{5}{3}Xija + Xijb)/30000\ for\ all\ i\ and\ j$$

$$ENTij = (\tfrac{5}{3}Xija + Xijb)/30000\ for\ all\ i\ and\ j$$

Objective Function:

Figure 3 - Hub Allocation Objective Function

$$Maximize \sum_{i=0}^{12}[\sum_{j=0}^{31} Xija * (Pa - Dij * Ta) + Xijb * (Pb - Dij * Tb) - NTij * 25000] - Mi * MCi$$

The output of this model gives the exact number of units for each product A and B shipped to each customer from each manufacturer, as well as the number of trucks required. Currently there is an assumption that the problem will not utilize milk runs or routing. Based on the production quantities, each customer likely requires a group trucks to transport all their quantities of A and B and these trucks will never travel between customers. Later in the project, the solution will be improved by implementing routes when possible. The next step is to find the cheapest way to assign the manufacturers a proper supplier selection.

4.1.2 - Supplier Selection (Supplier to Manufacturer)

Once the number of units shipped to each customer from all manufacturers is selected, a selection of what suppliers will supply which parts to the selected manufacturers must be completed. Due to the assumption that the transportation costs between supplier and manufacturer are included in the cost, it can be assumed that all parts are supplied to a central hub and then distributed across manufacturers. The model created for this part of the problem aims to minimize supplier costs.

Sets:

- I: Manufacturers
- J: Customers
- K: Products
- S: Suppliers
- α: Sub Item for A
- β: Sub Item for B

Parameters:

- Dijk: Demand supplied for k from manufacturer i to customer j
- CAsα: Capacity of a sub-item α for a supplier s
- CBsβ: Capacity of a sub-item β for a supplier s
- PAsα: Price of a sub-item α for a supplier s
- PBsβ: Price of a sub-item β for a supplier s

Variables:

- MSAsiα,: Sub-item α supplied to manufacturer i from supplier sj
- MSAsiβ,: Sub-item β supplied to manufacturer i from supplier sj

Constraints:

Figure 4 - Supplier Selection Constraints

$$\sum_{s=0}^{21} MSAsi\alpha = \sum_{j=0}^{31} Dija \; for \; all \; i \; and \; \alpha$$

$$\sum_{s=0}^{21} MSAsi\beta = \sum_{j=0}^{31} Dijb \; for \; all \; i \; and \; \beta$$

$$\sum_{i=0}^{12} MSAsi\alpha = CAs\alpha \; for \; all \; s \; and \; \alpha$$

$$\sum_{i=0}^{12} MSAsi\beta = CBs\beta \; for \; all \; s \; and \; \beta$$

Objective Function:

Figure 5 - Supplier Selection Objective Function

$$Minimize \sum_{\alpha}^{all} \sum_{i=0}^{12} \sum_{s=0}^{21} MSAsi\alpha * PAs\alpha + \sum_{\beta}^{all} \sum_{i=0}^{12} \sum_{s=0}^{21} MSAsi\beta * PAs\beta$$

The cost of supplying the manufacturers given each of the shipping strategies generated in step one is attained once the model is run to completion. The model is complete with a solution, but there are further improvements to be done as addressed later in this report.

5 - Solution

Table 1 - Total Profit for all Pricing Combinations

Product A Price	Product B Price	Customer Revenue	Supply Cost	Total
$4.60	$5.60	$2,685,329.39	$1,704,507.38	$980,822.02
$4.60	$5.80	$2,762,400.27	$1,708,848.00	$1,053,552.27
$4.60	$6.00	$2,652,245.63	$1,829,766.30	$822,479.33
$5.00	$5.60	$3,041,835.11	$1,705,874.65	$1,335,960.46
$5.00	$5.80	$2,945,182.62	$1,724,083.99	$1,221,098.62
$5.00	$6.00	$3,010,197.10	$1,829,241.40	$1,180,955.70
$5.20	$5.60	$3,215,625.88	$1,716,639.46	$1,498,986.42
$5.20	$5.80	$3,301,441.58	$1,720,268.99	$1,581,172.59
$5.20	$6.00	$3,000,237.71	$1,756,103.20	$1,244,134.51

After running the model with all 9 possible pricing combinations, the total profits for each maximized strategy are obtained. The pricing strategy that returned the best profit is assigning product A retail price of

$5.20 and product B a retail price of $5.80. The solution determined by the model activates manufacturing plants 1, 8, and 12 to satisfy the demand for both A and B. The production numbers for all the active manufacturing plants can be seen in table 2 in the “Grand Total” column for both product type A and product type B. The index numbers for the manufacturing plants are shifted to index zero, so manufacturing plant 0 in the table is manufacturing plant 1. The assignment of manufacturing plants to customer demand can be seen in table 2. In this table, the number of units shipped from manufacturer to customer of product type A or B can be seen. The “A Total” and “B Total” rows indicate the total demand of A and B satisfied for each of the customers.

Table 2 - Manufacturer-Customer Quantities

Units Assigned	Customer																															
Manufacturer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	23	24	25	26	27	28	29	30	31	Grand Total
A																																
0	45600		36000			27600		31800		30600	45600		34099			43200						17999			30600		16800		27600			387498
7		16372					19200		20700			27000	1900				40000		12600	26400	39000			26000		30000				19200		278372
11				21000	42000				16500					38000	43200			40200				31800	30000					26400			23400	312500
A Total	45600	16372	36000	21000	42000	27600	19200	31800	37200	30600	45600	27000	35999	38000	43200	43200	40000	40200	12600	26400	39000	49799	30000	26000	30600	30000	16800	26400	27600	19200	23400	978370
B																																
0	14000		30000			14000		7000		39000	14000		3168			48000									39000		32000		14000			254168
7		2712					28000		25500			44000	26832				23000		39000	46000	25000			46000		5000				28000		339044
11				25000	18000				2500					53000	48000			53000				37000	39000					16000			21000	312500
B Total	14000	2712	30000	25000	18000	14000	28000	7000	28000	39000	14000	44000	30000	53000	48000	48000	23000	53000	39000	46000	25000	37000	39000	46000	39000	5000	32000	16000	14000	28000	21000	905712
Grand Total	59600	19084	66000	46000	60000	41600	47200	38800	65200	69600	59600	71000	65999	91000	91200	91200	63000	93200	51600	72400	64000	86799	69000	72000	69600	35000	48800	42400	41600	47200	44400	1884082

As seen above, the demand of each customer is not necessarily satisfied by a single manufacturing plant. The number of trucks required to ship the units from manufacturing plant to customer can be seen in Appendix A. Table 3 is a summary of those numbers, displaying the number of trucks sent between each manufacturing plant and each customer.

Table 3 - Manufacturer-Customer Truck Allocation

Sum of Trucks	Customer																															
Manufacturer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	23	24	25	26	27	28	29	30	31	Grand Total
0	3		3			2		2		3	3		2			4						1			3		2		2			30
7		1					2		2			3	1				3		2	3	3			3		2				2		27
11				2	3				1					4	4			4				3	3					2			2	28
Grand Total	3	1	3	2	3	2	2	2	3	3	3	3	3	4	4	4	3	4	2	3	3	4	3	3	3	2	2	2	2	2	2	85

The results for the second portion of the project are the supplier-manufacturer relationships. The only suppliers selected to remain inactive in supplying subassemblies for product A are suppliers 6, 18, and 22; all other suppliers are active and supplying subassemblies to the manufacturing plants. The only suppliers selected to remain inactive in supplying subassemblies for product B are suppliers 1, 3, 11, 15, 18, 21,and 22 with all other suppliers being activated by the model. The total subassembly quantities relating to the production of product A and product B for each supplier can be seen in table 4 and table 5 respectively. The subassembly and supplier index numbers are shifted to index zero, so supplier number 1 is supplier 2 and subassembly 0 is subassembly 1.

Table 4 - Product A Supplier Subassembly Quantities

Quantity	Subassembly								
Supplier	0	1	2	3	4	5	6	7	Grand Total
0	320000					110000			430000
1					230000	340000			570000
2								260000	260000
3								210000	210000
4					100000				100000
6		120000							120000
7		400000							400000
8		18370				400000			418370
9		200000		200000					400000
10					400000				400000
11								158370	158370
12		240000		290000					530000
13	340000		150000		210000	8370		350000	1058370
14	18370								18370
15							950000		950000
16	300000		238370						538370
17				198370	38370		28370		265110
19			200000						200000
20			390000	290000		120000			800000
Grand Total	978370	978370	978370	978370	978370	978370	978370	978370	7826960

Table 5 - Product B Supplier Subassembly Quantities

Quantity	Subassembly							
Supplier	0	1	2	3	4	5	6	Grand Total
1	400000							400000
3			110000	270000				380000
4		250000					320000	570000
5				130000				130000
6	295712	300000	340000					935712
7				270000	230000			500000
8		280000			95712			375712
9	100000			15712	180000	500000		795712
11				220000	400000			620000
12						405712		405712
13			160000					160000
14		75712					270000	345712
16			105712				145712	251424
18			190000					190000
19	110000						170000	280000
Grand Total	905712	905712	905712	905712	905712	905712	905712	6339984

For more details on the product flows, truck requirements and subpart flow please refer to *"Appendix A: Hub Allocation Results"* and *"Appendix B: Supplier Selection Results"*.

6 - Reducing Fixed Cost of Transportation Manufacturing Plant to Customer

The objective function in the optimization model contains a fixed cost factor that account for the number of trucks departing from each manufacturing plant carrying final products to customers. Currently, how the number of trucks departing from each manufacturing plant is calculated as follows:

- Determine the total number of trucks required to transport the number of units of Product A and Product B decided to be sent to an individual customer from each manufacturing plant
- If the total number of trucks required is a decimal value, it is rounded up

This essentially results in partial truckloads being sent from one manufacturing plant to one customer, while the cost of using a truck to transport products is $25,000 regardless of its utilization. Hence, an attempt is taken to formulate an appropriate method to improve the utilization of trucks by combining partial truck loads to reduce the fixed costs of using trucks.

6.1 - Method

6.1.1 - Step 1: Determine Feasibility

Initially for each demand pair out of the 9 possible instances, the partial truckloads are added to see if the consolidating the partial truckloads will result in any savings. The criterion to determine feasibility is:

Number of partial truck load instances > Number of trucks required after consolidation

The number obtained after adding all the partial load values to be consolidated will be rounded up. This criterion will give an inference on whether the number of trucks required for transporting products to customers by consolidating their respective partial loads in milk runs, will be lesser than the number of trucks required determined in the original optimization model. If the criterion is not satisfied, fixed cost savings would not be realized by consolidating partial truckloads of demand and delivering them in milk runs. Table 2 lists the feasibility of applying transportation improvement for the 9 demand instances.

Table 6 - Feasibility Test of Transportation Improvement

Instance	Unit Price of Product A	Unit Price of Product B	No. of partial truck load instances	No. of trucks required after consolidation	Feasible to proceed (Yes/No)
1	$4.6	$5.6	4	3.43 ~ 4	No
2	$4.6	$5.8	3	2.84 ~ 3	No
3	$4.6	$6.0	14	12.01~13	Yes
4	$5.0	$5.6	11	10.61 ~ 11	No
5	$5.0	$5.8	6	5.78 ~ 6	No
6	$5.0	$6.0	21	14.99 ~ 15	Yes
7	$5.2	$5.6	13	12.32 ~ 13	No
8	$5.2	$5.8	11	10.54 ~ 11	No
9	$5.2	$6.0	26	17.06 ~ 18	Yes

6.1.2 - Step 2: Identify manufacturing plants which send partial truckloads to multiple customers

Consolidating partial truckloads originating from one manufacturing plant ensures lower variable cost of transporting units of products on the truck, from each identified feasible instances from step 1 above. This is achieved due to the elimination of travel between manufacturing plants. In this step as shown in Table 3, the occurrences of one manufacturing plants sending partial truckloads to multiple customers is identified, in each of the feasible demand instances identified in Step 1 above.

Table 7 – Partial Truckload Instances

Instance 3			Instance 6			Instance 9		
Plant	Customer	Partial Load	Plant	Customer	Partial Load	Plant	Customer	Partial Load
1	6	0.733	1	28	0.489	1	28	0.433
1	3	0.6	1	6	0.4	1	6	0.289
1	13	0.99	1	30	0.956	1	30	0.9
8	13	0.99	1	1	0.733	1	1	0.678
8	25	0.9	1	11	0.844	1	8	0.956
8	17	0.911	1	26	0.622	1	11	0.622
8	20	0.844	8	25	0.733	1	26	0.4
8	2	0.99	8	17	0.8	8	25	0.678
8	12	0.844	8	27	0.956	8	17	0.689

12	5	0.911	8	20	0.622	8	27	0.733
12	15	0.7	8	9	0.445	8	20	0.511
12	22	0.976	8	19	0.956	8	9	0.505
12	32	0.967	8	12	0.789	8	19	0.733
12	14	0.622	8	31	0.589	8	12	0.567
			12	5	0.689	8	21	0.933
			12	15	0.533	8	31	0.422
			12	9	0.99	12	23	0.9
			12	29	0.467	12	5	0.633
			12	32	0.856	12	15	0.478
			12	14	0.567	12	9	0.884
						Instance		
						Plant	Customer	Partial Load
						12	29	0.411
						12	32	0.633
						12	4	0.933
						12	14	0.511
						12	24	0.833
						12	18	0.789

6.1.3 - Step 3: Identify feasible combinations partial truckload to multiple customers from one manufacturing plant and route

Consolidating partial truckloads from the identified opportunities from step 2, will materialise when the combined volume of the partial truckloads selected to be sent on a milk run can be contained in one truck. The criteria to determine feasibility of partial truckload combinations are explained below.

- Sum of partial truck load instances < One full truckload
- Prioritize combining partial truckload demands of customers located minimum distance apart
- Deliver to customer with larger volume demand of partial in the milk run before customer with smaller volume demand

By combining partial truckload demands of customers located closer to each other and delivering larger volume demands first, ensures lower variable cost of carrying products on the truck. This is due to the fact that lesser number of products are carried on the truck by applying those criteria. Table 4 lists the feasible milkruns applicable for Instance 6 and Instance 9.

Table 8 - Feasible Milkruns from Manufacturer to Multiple Customers

Instance 6			Instance 9		
Plant	Customers served	Milk Run Route	Plant	Customer	Partial Load
1	28, 6	P1-C28-C6-P1	1	28, 26	P1-C28-C26-P1
12	15, 29	P12-C15-C29-P15	1	1, 6	P1-C1-C6-P1
			8	9, 31	P8-C9-C31-P8
			12	14, 29	P12-C14-C29-P12

6.1.4 - Step 4: Calculate Fixed Cost Savings

The cost saving is achieved from the reduction of number of trucks used by the improved routing of partial truckloads of demands compared to the original solution.

Figure 6 - Fixed Cost Savings

$$Fixed\ Cost\ Saving = (\#\ of\ trucks\ used\ initially - \#\ of\ trucks\ used\ after\ improvement) * \$25000$$

The results after applying the improvement method for Instances 6 and 9 are listed in Table 5. Number of trucks required to deliver products from the manufacturing plants to customers has reduced by two in instance 6, saving $50,000. And, in instance 9, four less trucks are required resulting in a $100,000 saving.

Table 9 - Truck Requirement Results

Instance 6			
Milk Run Route	No. of trucks required in solution	No. of trucks required after improvement	Reduced Truck Requirement
P1-C28-C6-P1	2	1	1
P12-C15-C29-P15	2	1	1
Instance 9			
Milk Run Route	No. of trucks required in solution	No. of trucks required after improvement	Reduced Truck Requirement
P1-C28-C26-P1	2	1	1
P1-C1-C6-P1	2	1	1
P8-C9-C31-P8	2	1	1
P12-C14-C29-P12	2	1	1

6.1.5 - Step 5: Cost Savings Impact

Table 6 shows that the profit increases in both price combinations after savings improvement doesn't result in either of them becoming the optimal profit generating price levels. Hence, the profit maximizing price level is still, $5.20 for product A retail price and $5.80 for product B.

Table 6 - Total Profit for after Transportation Improvement

Instance	Product A Price	Product B Price	Profit Before Improvement	Savings from Improvement	Profit After Improvement
6	$5.00	$6.00	$1,180,955.70	$50,000	$1,230,955.70
9	$5.20	$6.00	$1,244,134.51	$100,000	$1,344,134.51

7 - Conclusion

In summary of the findings of this project, based on the assumptions made in creating the solution methodologies and model for this project and the constraints provided in the problem description, the supply chain discussed in section 5 provides the most optimal profit of $1,581,172.59. This supply chain is developed by breaking the problem into two portions, a hub allocation problem and a supplier allocation problem that must be solved in sequence. The hub allocation problem accounts for assigning customer demand to production facilities based on transportation costs and production prices. The supplier allocation problem accounts for assigning subassembly suppliers to manufacturing plants based on the cost and capacity of each supplier.

The solution presented in section 5 is optimal but the solution method can be improved to account for the suboptimal transportation of partial truckloads that can occur using this model. The optimal supply chain does not benefit from utilizing transportation routing improvements between the manufacturer and the customer but depending on the shipment quantities, the transportation cost of other supply chains can be improved. This improvement can be made by implementing a milk run method to combine partially full shipments to different customers from a manufacturer. In section 6 of this report the impact of implementing a milk run method on one of the less optimal supply chain solutions created by the model is analyzed. It is shown that implementing a routing strategy can result in a savings of up to $100 000 based on the circumstance.

Appendix A: Hub Allocation Results

Table 10 - Shipment Flows for Optimal Pricing

Manufactory	Customer	Product	Units Shipped
11	31	B	21000
0	5	A	27600
7	20	A	39000
11	28	A	26400
0	12	B	3168
0	7	A	31800
11	13	B	53000
11	4	B	18000
11	14	B	48000
7	6	B	28000
7	8	B	25500
11	22	B	37000
0	9	A	30600
7	11	A	27000
11	8	B	2500
0	2	B	30000
7	16	A	40000
0	10	A	45600
0	9	B	39000
7	6	A	19200
11	14	A	43200
0	10	B	14000
0	25	A	30600
11	13	A	38000
7	20	B	25000
11	3	A	21000
11	23	A	30000
0	0	B	14000
7	18	B	39000
7	1	B	2712
0	27	A	16800
0	15	A	43200
0	27	B	32000
7	30	A	19200
0	5	B	14000
0	15	B	48000
7	30	B	28000
0	29	A	27600
7	11	B	44000
7	26	B	5000
7	18	A	12600
7	12	A	1900

7	24	A	26000
0	22	A	17999
0	25	B	39000
7	19	B	46000
7	19	A	26400
0	12	A	34099
7	16	B	23000
7	1	A	16372
11	31	A	23400
11	22	A	31800
7	8	A	20700
11	8	A	16500
11	17	A	40200
7	12	B	26832
11	3	B	25000
0	7	B	7000
0	0	A	45600
7	24	B	46000
11	23	B	39000
11	17	B	53000
11	28	B	16000
11	4	A	42000
7	26	A	30000
0	29	B	14000
0	2	A	36000

Table 11 - Trucks Required for Optimal Pricing

Manufactory	Customer	Trucks
7	12	1
11	22	3
0	27	2
7	24	3
11	4	3
0	5	2
11	14	4
0	15	4
11	8	1
0	9	3
7	16	3
0	29	2
0	0	3
7	26	2
11	28	2
0	7	2
0	10	3
7	6	2
7	19	3

7	8	2
11	31	2
7	1	1
7	18	2
11	3	2
0	2	3
7	11	3
7	20	3
11	13	4
0	22	1
0	25	3
0	12	2
7	30	2
11	23	3
11	17	4

Appendix B: Supplier Selection Results

Table 12 - Flow of Sub-Parts for Item A Under Optimal Pricing

Supplier	Manufacturer	Sub-Part	Quantities
17	0	6	28370
13	0	2	71628
7	7	1	158372
8	0	1	18370
20	0	2	77500
15	11	6	312500
13	7	4	210000
15	7	6	278372
13	0	0	49128
4	0	4	100000
20	11	2	312500
20	11	5	120000
13	11	0	290872
15	0	6	359128
13	7	2	78372
2	11	7	260000
6	7	1	120000
17	0	3	198370
13	0	7	19128
12	0	3	11628
10	11	4	274130
10	7	4	68372
8	11	5	20872
11	0	7	158370
13	7	7	278372
13	0	5	8370
0	0	0	320000

12	0	1	127500
20	0	3	177500
20	11	3	112500
16	7	0	278372
10	0	4	57498
0	11	5	110000
16	0	2	238370
14	0	0	18370
17	11	4	38370
9	11	3	200000
3	0	7	210000
13	11	7	52500
7	0	1	241628
8	0	5	379128
1	11	5	61628
1	7	5	278372
1	0	4	230000
9	11	1	200000
16	11	0	21628
12	11	1	112500
19	7	2	200000
12	7	3	278372

Table 13 - Flow of Sub-Parts for Item B Under Optimal Pricing

Supplier	Manufacturer	Sub-Part	Quantities
5	11	3	92500
3	7	3	31544
6	0	0	254168
1	7	0	297500
8	11	1	280000
7	7	3	270000
9	0	3	15712
9	11	4	180000
11	0	4	158456
13	7	2	160000
18	11	2	190000
6	0	2	144168
6	7	1	89044
12	7	5	151544
19	11	6	170000
8	0	4	95712
6	11	2	122500
16	0	6	145712
11	7	4	109044
11	11	3	220000
11	11	4	132500

16	7	2	105712
9	11	0	100000
4	11	6	142500
3	0	3	238456
4	7	6	177500
5	7	3	37500
6	0	1	178456
9	11	5	312500
1	11	0	102500
14	0	6	108456
14	0	1	75712
6	7	0	41544
12	0	5	254168
14	7	6	161544
6	11	1	32500
7	7	4	230000
6	7	2	73332
3	0	2	110000
4	7	1	250000
19	11	0	110000
9	7	5	187500

Appendix C: Python Code for Gurobi Model – Hub Allocation

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pyomo.core.base import ConcreteModel
from pyomo.environ import *
import math
import pandas as pd

# Creation of a Concrete Model
model = ConcreteModel()

## Define Sets ##
#  Sets
#   i   Manufacturers
#   j   Customers
#   k   Products

model.i = Set(initialize=range(0, 13), doc='Manufacturers')
model.j = Set(initialize=range(0, 32), doc='Customers')
model.k = Set(initialize=['A', 'B'], doc='Sources')

## Define Parameters ##
#  Parameters
#   Ri      Running costs for manufacturer
#   Dij     Distance from manufacturer to customer
#   Tk      Transport cost of product k
#   CDjk    Demand of product k for customer j
#   Pk      Selling price of k
#   TCk     Truck capacity for k

distances = pd.read_csv("distances.csv", header=None).T.to_dict()
D = {}
for j in range(0, 32):
    for i in range(0, 13):
```

```
            D[(i, j)] = distances[j][i]
    model.D = Param(model.i, model.j, initialize=D, doc='Distances')

    demands = pd.read_csv("demands.csv", header=None).T.to_dict()
    CD = {}
    for j in range(0, 32):
        for k in ['A', 'B']:
            if k == 'A':
                CD[(j, k)] = demands[j][2]*1000
            if k == 'B':
                CD[(j, k)] = demands[j][5]*1000
    model.CD = Param(model.j, model.k, initialize=CD, doc='Demands')
    MC = {
        0: 1404000,
        1: 1452000,
        2: 1476000,
        3: 1482000,
        4: 1500000,
        5: 1488000,
        6: 1434000,
        7: 1356000,
        8: 1464000,
        9: 1470000,
        10: 1458000,
        11: 1392000,
        12: 1392000
    }
    model.MC = Param(model.i, initialize=MC, doc='Running Costs')

    model.T = Param(model.k, initialize={'A': 0.0015, 'B': 0.0020}, doc='Transport Costs')
    model.TC = Param(model.k, initialize={'A': 50000, 'B': 30000}, doc='Truck Costs')
    model.P = Param(model.k, initialize={'A': 5.2, 'B': 6.0}, doc='Product Price')
    model.R = Param(model.k, initialize={'A': 35, 'B': 45}, doc="Hours of Work")
    model.UL = Param(model.k, initialize={'A': 1280000, 'B': 940000}, doc="Upper Limit")

    ## Define variables ##
    model.X = Var(model.i, model.j, model.k, within=NonNegativeIntegers, doc="Product k flow from i to j")
    model.M = Var(model.i, within=Binary, doc="Manufacturing plant i exists")
    model.NT = Var(model.i, model.j, within=NonNegativeIntegers, doc="Trucks Required")
    model.ENT = Var(model.i, model.j, within=NonNegativeReals, doc="Exact Trucks Required")

    ## Define constraints ##
    def Demand(model, j, k):
        return sum(model.X[i, j, k] for i in model.i) <= model.CD[j, k]
    model.Demand = Constraint(model.j, model.k, rule=Demand, doc='Supply Demand')

    def DemandLimits(model, k):
        return sum(model.X[i, j, k] for i in model.i for j in model.j) <= model.UL[k]
    model.DemandLimits = Constraint(model.k, rule=DemandLimits, doc='Upper Limit Supplier Demand')

    def Capacity(model, i):
        return sum(model.X[i, j, k]*(model.R[k]/1000) for j in model.j for k in model.k) <= 25000*model.M[i]
    model.Capacity = Constraint(model.i, rule=Capacity, doc='Manufacturer Capacity')

    def Trucks(model, i, j):
        return model.NT[i, j] >= ((5/3)*model.X[i, j, 'A'] + model.X[i, j, 'B'])/30000
    model.Trucks = Constraint(model.i, model.j, rule=Trucks, doc='Trucks Required')

    def RealTrucks(model, i, j):
        return model.ENT[i, j] == ((5/3)*model.X[i, j, 'A'] + model.X[i, j, 'B'])/30000
    model.RealTrucks = Constraint(model.i, model.j, rule=RealTrucks, doc=' Exact Trucks Required')


    ## Define Objective and solve ##
    def objectiveRule(model):

```

```
    return sum(sum(model.X[i, j, 'A']*(model.P['A'] - model.D[i, j]*model.T['A']) + model.X[i, j, 'B']*(model.P['B'] - model.D[i, j]*model.T['B']) - model.NT[i, j]*25000 for j in model.j) - model.M[i]*model.MC[i] for i in model.i)
model.objectiveRule = Objective(rule=objectiveRule, sense=maximize, doc='Define Objective Function')


def pyomo_postprocess(options=None, instance=None, results=None):
  with open('Manufacturer.txt', 'w') as f:
      for key, value in instance.M._data.items():
          if value._value > 0:
            f.write('%s,%s\n' % (key, value._value))
  with open('Flow of Goods.txt', 'w') as f:
      for key, value in instance.X._data.items():
          f.write('%s,%s,%s,%s\n' % (key[0], key[1], key[2], value._value))
  with open('Number Of Trucks.txt', 'w') as f:
      for key, value in instance.NT._data.items():
          if value._value > 0:
            f.write('%s,%s,%s\n' % (key[0], key[1], value._value))
  with open('Exact Number Of Trucks.txt', 'w') as f:
      for key, value in instance.ENT._data.items():
          if value._value > 0:
            f.write('%s,%s,%s\n' % (key[0], key[1], value._value))


if __name__ == '__main__':
    # This emulates what the pyomo command-line tools does
    from pyomo.opt import SolverFactory
    import pyomo.environ

    opt = SolverFactory("gurobi")
    results = opt.solve(model)
    # sends results to stdout
    results.write()
    print("\nWriting Solution\n" + '-' * 60)
    pyomo_postprocess(None, model, results)
    print("Complete")
```

Appendix C: Python Code for Gurobi Model – Supplier Selection

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pyomo.core.base import ConcreteModel
from pyomo.environ import *
import math
import pandas as pd

# Creation of a Concrete Model
model = ConcreteModel()

## Define Sets ##
#  Sets
#   i   Manufacturers
#   j   Customers
#   k   Products

model.i = Set(initialize=range(0, 13), doc='Manufacturers')
model.j = Set(initialize=range(0, 32), doc='Customers')
model.s = Set(initialize=range(0, 22), doc='Suppliers')
model.k = Set(initialize=['A', 'B'], doc='Sources')
model.sub_a = Set(initialize=range(0, 8), doc='Items for Product A')
model.sub_b = Set(initialize=range(0, 7), doc='Items for Product B')

## Define Parameters ##
#  Parameters

D = {}
j = []
i = []
demands = pd.read_csv("Flow of Goods.csv", header=None).T.to_dict()
for key in demands:
    D[(demands[key][0], demands[key][1], demands[key][2])] = int(demands[key][3])
model.D = Param(model.i, model.j, model.k, initialize=D, doc='Manufacturer Demands')

CA = {}
capacities_a = pd.read_csv("capacity-a.csv", header=None).T.to_dict()
for s in range(0, 22):
    for c in range(0, 8):
        if capacities_a[s][c] != '-':
            CA[(s, c)] = int(capacities_a[s][c])*1000
        else:
            CA[(s, c)] = 0
model.CA = Param(model.s, model.sub_a, initialize=CA, doc='Sub Item Capacities for A')

CB = {}
capacities_b = pd.read_csv("capacity-b.csv", header=None).T.to_dict()
for s in range(0, 22):
    for c in range(0, 7):
        if capacities_b[s][c] != '-':
            CB[(s, c)] = int(capacities_b[s][c])*1000
        else:
            CB[(s, c)] = 0
model.CB = Param(model.s, model.sub_b, initialize=CB, doc='Sub Item Capacities for B')

PA = {}
pricing_a = pd.read_csv("pricing-a.csv", header=None).T.to_dict()
for s in range(0, 22):
    for c in range(0, 8):
        if pricing_a[s][c] != '-':
            PA[(s, c)] = float(pricing_a[s][c])
        else:
            PA[(s, c)] = 0
model.PA = Param(model.s, model.sub_a, initialize=PA, doc='Sub Item Pricing for A')

PB = {}
pricing_b = pd.read_csv("pricing-b.csv", header=None).T.to_dict()
```

```python
for s in range(0, 22):
    for c in range(0, 7):
        if pricing_b[s][c] != '-':
            PB[(s, c)] = float(pricing_b[s][c])
        else:
            PB[(s, c)] = 0
model.PB = Param(model.s, model.sub_b, initialize=PB, doc='Sub Item Pricing for B')

## Define variables ##
model.MSA = Var(model.s, model.i, model.sub_a, within=NonNegativeIntegers, doc="Sub Item A flow from s to i")
model.MSB = Var(model.s, model.i, model.sub_b, within=NonNegativeIntegers, doc="Sub Item B flow from s to i")

## Define constraints ##
def SubDemandA(model, i, sub_a):
    return sum(model.MSA[s, i, sub_a] for s in model.s) == sum(model.D[i, j, 'A'] for j in model.j)
model.SubDemandA = Constraint(model.i, model.sub_a, rule=SubDemandA, doc='Sub Items for A')

def SubDemandB(model, i, sub_b):
    return sum(model.MSB[s, i, sub_b] for s in model.s) == sum(model.D[i, j, 'B'] for j in model.j)
model.SubDemandB = Constraint(model.i, model.sub_b, rule=SubDemandB, doc='Sub Items for B')

def SupplierCapacityA(model, s, sub_a):
    return sum(model.MSA[s, i, sub_a] for i in model.i) <= model.CA[s, sub_a]
model.SupplierCapacityA = Constraint(model.s, model.sub_a, rule=SupplierCapacityA, doc='Supplier Capacities for A')

def SupplierCapacityB(model, s, sub_b):
    return sum(model.MSB[s, i, sub_b] for i in model.i) <= model.CB[s, sub_b]
model.SupplierCapacityB = Constraint(model.s, model.sub_b, rule=SupplierCapacityB, doc='Supplier Capacities for B')


## Define Objective and solve ##
def objectiveRule(model):

    return sum(sum(sum(model.MSA[s, i, sub_a]*model.PA[s, sub_a] for sub_a in model.sub_a) for i in model.i) for s in model.s) + sum(sum(sum(model.MSB[s, i, sub_b]*model.PB[s, sub_b] for sub_b in model.sub_b) for i in model.i) for s in model.s)
model.objectiveRule = Objective(rule=objectiveRule, sense=minimize, doc='Define Objective Function')


def pyomo_postprocess(options=None, instance=None, results=None):
  with open('Supplier Flows - A.txt', 'w') as f:
      for key, value in instance.MSA._data.items():
          if value._value > 0:
            f.write('%s,%s,%s,%s\n' % (key[0], key[1], key[2], value._value))
  with open('Supplier Flows - B.txt', 'w') as f:
      for key, value in instance.MSB._data.items():
          if value._value > 0:
            f.write('%s,%s,%s,%s\n' % (key[0], key[1], key[2], value._value))


if __name__ == '__main__':
    # This emulates what the pyomo command-line tools does
    from pyomo.opt import SolverFactory
    import pyomo.environ

    opt = SolverFactory("gurobi")
    results = opt.solve(model)
    # sends results to stdout
    results.write()
    print("\nWriting Solution\n" + '-' * 60)
    pyomo_postprocess(None, model, results)
    print("Complete")
```