Minimizing Retail Costs through Cooperation in a Supply Chain

Authors: Ryan Jaipersaud, Armaan Thapar
Instructor: Professor Benjamin Davis

ChE-488 Convex Optimization
Cooper Union for the Advancement of Science and Art
Date of submission 12/19/18

Abstract

It is in the interest of any company that sells a product to minimize shipping, ordering, and storage costs. One factor in reducing costs is to find an optimal replenishment strategy to meet market demand. We explore here the difference between a clothing retailer acting independently versus cooperating with its supplier in a two-chain supply management problem. In this paper, a startup retailer and supplier selling shirts negotiate their optimal replenishment strategy. Acting independently, the total cost for the retailer and supplier is \$11,142 and \$7,632.00, respectively. However, if the two companies work together the costs are \$11,542 and \$6,378, respectively. The supplier can then offer a side payment of \$400 dollars to cover the increase in the retailers cost while decreasing its own by \$854 overall with the negotiated strategy. The optimization problem discussed is a binary integer linear problem since the retailer and supplier have the option to place or not to place an order each month. In order to solve for the optimal replenishment plan we developed a program that reduces the BILP to an LP and solves the LP over all combinations of ordering plans. A comparison between the independent case and the cooperation case was performed. PuLP, a python based solver is used to solve the LPs. For a time period of 5 months, 64 LPs were solved for the independent case and 1024 LPs were solved in the cooperation case. Future work would include generalizing the optimization setup and program to account for other goods such as food which are perishable as well as discounts from buying in bulk.

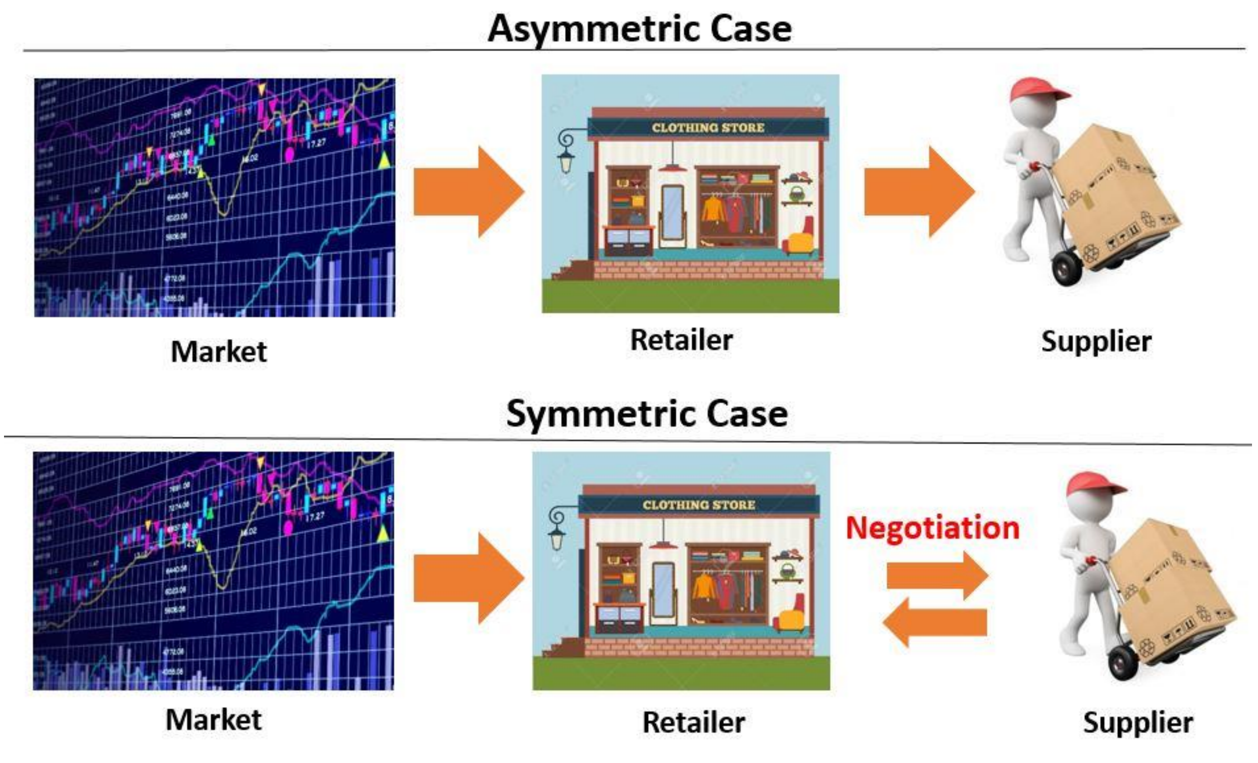

Diagram 1: Exchange of Information in Symmetric and Asymmetric Cases[i]

Introduction

In supply chain management, finding the optimal replenishment plan, which is the amount of product to buy and hold over the course of a time period, proves to be a difficult problem. As the time period increases the number of possible plans increases significantly. Future demand can affect storage costs for prior months. The cost of a product is tied to shipping, ordering units and holding units in storage. For many companies, inventory is the largest asset owned by the company, and these companies need to carry enough product to meet market demands.[ii] If a company orders too much at the beginning of a year, the storage costs will eat up a large portion of the profits, and if the company orders too little, then there may not be enough product to meet demand. This would result in a missed sale. To minimize storage costs, one may try to make frequent orders but, then that would increase the shipping costs of the product. Other factors that come into play are limits on the amount of product one can order at any given time as well as the amount one can hold in storage. Thus, it is important for the company to minimize its cost by attempting to meet demand exactly. In this report demand is assumed to vary in time, or else the problem would reduce to the economic order quantity problem where there exists a closed form solution.

Diagram 2 shows how goods and demands move through a supply chain. On the left side of the diagram the Retailer and Supplier are working separately. The supplier only knows the demands of the retailer and nothing more. The argument is that by working together (right side of the diagram), where the supplier knows the market demand set by the consumer and constraints on the retailer, the supplier can propose a replenishment plan to the retailer such that costs are lower for both parties.

It is important to note the retailer and supplier are actors with competing objectives and thus information is usually not shared between the two. In fact, while the retailer has to meet the market demand, the supplier needs to meet the retailer's demand. This is true for large retailers such as Walmart and food stores where the retailer has more information of the market demand, and the supplier is waiting on the retailer to know how to optimize its own replenishment plan.[iii] However, the replenishment strategy forced by the retailer can prove unfavorable to the supplier.

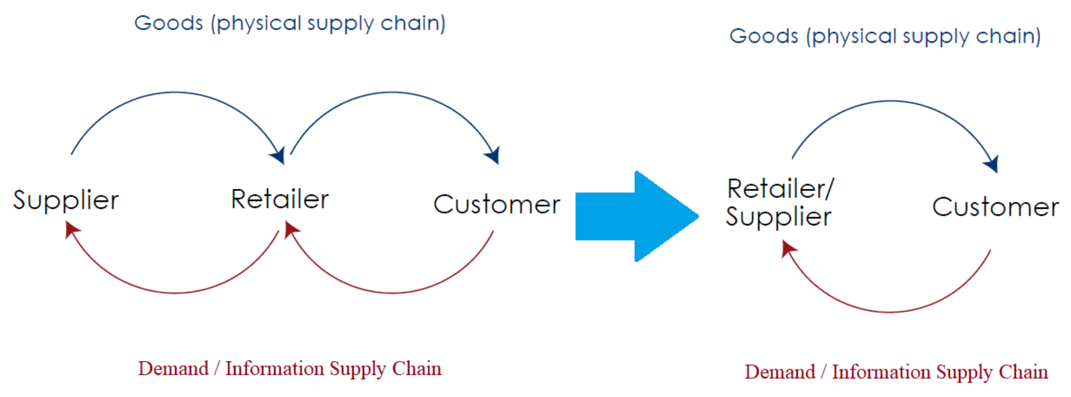
Goods (physical supply chain)
Supplier
Retailer
Customer
Demand / Information Supply Chain
Goods (physical supply chain)
Retailer/ Supplier
Customer
Demand / Information Supply Chain

Diagram 2: Flow of Goods and Information in a Supply Chain[iv]

<u>Definition of variables</u>

$$\mathrm{t} = \text{time index}$$
$$\mathrm{T} = \text{stop time}$$
$$\tau = \{1,2,\ldots \mathrm{T}\} = \text{space of all time to place an order}$$
$$\mathrm{x_t} = \text{units to be ordered } \textit{at time t}$$
$$\mathrm{y_t} = \text{placing an order at time t}$$
$$\mathrm{s_t} = \text{units in storage at time t}$$
$$\mathrm{d_{tT}} = \text{cumulative demand from t to T} = \sum_{i=t}^{T} d_i$$
$$\mathrm{d_i} = \text{market demand at time i}$$

Formalized in scalar form in optimization problem A, all companies aim to minimize the cost function $\mathrm{C}(x, y, s)$. There are $3T$ decision variables ($x_t, y_t, s_t \quad \forall t \in \{1 \ldots T\}$) which grow as a function of T, affecting the complexity of the program. The time increment for this paper is in months. Companies have the ability to place an order (y_t) and then decided how much to order (x_t) to meet demand for each month t. They are allowed to use the product that they have stored from the previous month (s_{t-1}) to help meet the demand of the next month (t). Product that was not sold at time t is then put into storage for that month (s_t).

It is the company's job to determine all $3T$ variables to minimize the cost. The optimization problem requires solving 2^T LPs. Constraint 1 ensures demand is met. Constraint 2 is an upper limit to how many units need to be ordered at time t. This should always be less than the cumulative demand from t to T. There usually exists an order constraint, X, that places a tighter bound on the amount one can order. Constraint 3 is a positivity constraint to prevent the solver from ordering and holding negative units to minimize the objective function. Constraint 4 is a limit on the amount of storage. Note that d, S, X and T are parameters.

<u>Optimization Problem A</u>

$$\min \qquad \sum_{t=1}^{T} C(x_t, y_t, s_t)$$

$$\text{s.t.} \qquad s_{t-1} + y_t x_t = d_t + s_t \quad \forall t \in \{1 \ldots T\}, \quad (1)$$

$$x_t \leq X \leq d_{tT} y_t \qquad \forall t \in \{1 \ldots T\}, \quad (2)$$

$$x_t, s_t \geq 0 \quad \forall t \in \{1 \ldots T\}, \quad (3)$$

$$s_t \leq S \quad \forall t \in \{1 \ldots T\}, (4)$$

$$y_t \in \{0,1\} \;\; \forall t \in \{1 \ldots T\}$$

$$x_t, s_t \in \mathbb{R} \; \forall t \in \{1 \ldots T\}$$

The motivation behind this study is to show that cooperation between the retailer and the supplier would result in a lower cost to both parties which is equivalent to solving optimization problem B taken from Phouratsamay et al.[v] In optimization problem B superscripts denote the company to which variables are assigned. The difference between A and B is in the number of variables that need to be solved, the objective function, and the number of constraints. Since the objective function now accounts for both the retailer and the supplier the solution that minimizes the value will reflect the overall cost to both parties together rather than each one individually. In this case there are $6T$ variables to be minimized over 2^{2T} LPs. For equation 2 and constraint 4, the orders from the retailer,x_t^R, can be thought of as the demand the supplier needs to meet. Note that it is the job of the supplier to solve this problem and to propose the replenishment plan.

Optimization Problem B

$$\min \qquad \sum_{t=1}^{T} C^R(x_t^R, y_t^R, s_t^R) + C^S(x_t^S, y_t^S, s_t^S)$$

$$\text{s.t.} \qquad s_{t-1}^R + x_t^R = d_t + s_t^R \;\; \forall t \in \{1 \ldots T\}, \qquad (1)$$

$$s_{t-1}^S + x_t^S = x_t^R + s_t^S \;\; \forall t \in \{1 \ldots T\}, \qquad (2)$$

$$x_t^R \le X^R \le \; d_{tT} y_t^R \;\; \forall t \in \{1 \ldots T\}, \qquad (3)$$

$$x_t^S \le \; X^S \le \; x_{tT}^R y_t^S \;\; \forall t \in \{1 \ldots T\}, \qquad (4)$$

$$s_t^R \le \; S^R \;\; \forall t \in \{1 \ldots T\}, \qquad (5)$$

$$s_t^S \le \; S^S \; \forall t \in \{1 \ldots T\}, \qquad (6)$$

$$x_t^R, s_t^R, x_t^S, s_t^S \ge 0 \;\; \forall t \in \{1 \ldots T\}, \qquad (7)$$

$$s_0^R, s_0^S = 0 \qquad (8)$$

$$y_t^R, y_t^S \epsilon \, \{0,1\}; \; x_t^R, x_t^S, s_t^R, s_t^S \epsilon \, \mathbb{R} \; \forall t \in \{1 \ldots T\},$$

If the cost the retailer faces under the supplier's solution to problem B is lower than under its own solution to problem A then the retailer will accept the plan. If the cost is not lower, the supplier can offer a side payment that will be equal to the offset in cost and will still potentially lower costs for both parties. Thus, we are trying to find the replenishment plan that would decrease costs under symmetric information. In other words, the optimization problem can be written as follows:

$$\begin{aligned}
&\text{min} && \text{Ordering and Storage costs for the supplier and the retailer}\\
&\text{s.t.} && \text{1.) The retailer meets market demand}\\
& && \text{2.) The supplier meets the retailer's demand}\\
& && \text{3.) The retailer cannot order more than the order limit}\\
& && \text{4.) The supplier cannot order more than the order limit}\\
& && \text{5.) The retailer cannot surpass the storage limit}\\
& && \text{6.) The supplier cannot not surpass the storage limit}\\
& && \text{7.) The amount of units ordered and stored is greater than or equal to 0}
\end{aligned}$$

Problem Description

A startup retailer in New York City is working with a startup t-shirt company in Philadelphia that supplies customizable t-shirts where you can add your own design to the shirt. Prices for shipping and ordering have been fixed and both companies are currently negotiating. The retailer has obtained market data from which to determine orders for the next 5 months. The supplier notifies the retailer that they can only produce 115 shirts a month and has an ordering cap of 110 shirts a month from the company prior in the chain. The retailer and the supplier can only hold up to (55 and 60 shirts) a month in storage, respectively. The supplier buys shirts for \$12 per shirt and sells them at \$20 per shirt. The inventory cost for the retailer and the supplier are \$8 and \$7 per shirt, respectively. The shipping cost for the retailer and the supplier is \$50 and \$30 per order, respectively. The objective is to find the best replenishment plan that minimizes the cost to both actors. Therefore the parameters are as follows: $\tau = \{1,2,3,4,5\},\ T = 5, S^R = 55,\ S^S = 60, X^R = 115, X^S = 110$.

To lower the cost to the retailer and the supplier, the general steps to solve the optimization problem are as follows:

1.) Determine parameters for the cost function for the retailer and supplier (shipping cost, ordering price per unit of product, storage cost per unit of product)
2.) Determine parameters for the constraints and equalities for the retailer and supplier (the market demand for the specific product, the limit on the units one can have in storage, the limit on the units that one can order)
3.) Setup Optimization A for both the retailer and the supplier separately
4.) Setup Optimization B for both the retailer and the supplier together
5.) Solve for the replenish strategy for each actor in both cases using PuLP[vi]
6.) If the replenishment strategy generated by B increases cost for the retailer then determine the side payment the supplier will make to the retailer.

Data was obtained from a previous study that looked into retail demand for clothing over 222 stores in 2007.[vii] The expected demand generated in Table 1 is from averaging the shirt sales in the first 5 months of 2007 over 222 retailers to find the average sales of each retailer. 5 months of analysis were chosen to create a model with sufficient complexity, but also one that converges quickly enough to perform multiple runs for sensitivity analysis. The price of a shirt was taken from the JCPenny's website and found by averaging over the shirt prices on the website.[viii] Shipping costs from Philadelphia to New York were taken from Fedex's website.[ix] Storage and ordering constraints were artificially created.

Average	**Demand**
January	54
February	103
March	116
April	105
May	141

Table 1: Expected Demand for Months

The positivity constraints on x_t^R, s_t^R, x_t^S, s_t^S are to prevent the linear programming solver from ordering negative units and holding negative quantities. The optimal value of the objective function would otherwise approach negative infinity and the problem would be unbounded.

The upper constraints on $s_t^R \leq 55$, $s_t^S \leq 60$ are to model real-world storage constraints. Storage units have a cap on how much one is allowed to store. The upper constraints on $x_t^R \leq 115$, $x_t^S \leq 110$ are to model ordering constraints. Companies cannot expect to order 10,000 units of product from a factory that only has 5 workers. There is a production limit on the company previously in the supply chain. These two constraints are coupled since in order to meet demand for a month yet still stay below the production constraint a company may have to order more the prior month and place the product in storage.

Because both actors have an option to place or not to place and order for each month $y_t \epsilon \{0,1\}$ the objective function is not convex. In order to figure out the optimal replenishment strategy we treated y_t^S and y_t^R as parameters and worked through all possible combinations. This reduces the BILP to an LP. The solver will keep solutions that are feasible to the LPs.

There are many ways to solve an LP such as with a simplex or an interior point method. This problem was solved by setting up optimization B in simplified form and using the PuLP library in Python to find the optimal replenishment plan. In this problem the method LpProblem() was used. Based on the scaling of the problem the solution is sufficient to model two small companies. Larger time periods can be solved, though it is combinatorially expensive.

<u>Problem Formulation</u>

$$(BILP) = \min \quad \sum_{t=1}^{T=5} 50y_t^R + 20x_t^R + 8s_t^R + 30y_t^S + 12x_t^S + 11s_t^S$$

$$\text{s.t.}$$

$$s_0^R + x_1^R = 54 + s_1^R \quad (1)$$
$$s_1^R + x_2^R = 103 + s_2^R \quad (2)$$
$$s_2^R + x_3^R = 116 + s_3^R \quad (3)$$
$$s_3^R + x_4^R = 105 + s_4^R \quad (4)$$
$$s_4^R + x_5^R = 141 + s_5^R \quad (5)$$
$$s_0^S + x_1^S = x_1^R + s_1^S \quad (6)$$
$$s_1^S + x_2^S = x_2^R + s_2^S \quad (7)$$
$$s_2^S + x_3^S = x_3^R + s_3^S \quad (8)$$
$$s_3^S + x_4^S = x_4^R + s_4^S \quad (9)$$
$$s_4^S + x_5^S = x_5^R + s_5^S \quad (10)$$
$$s_0^R, s_0^S = 0 \qquad (11)$$
$$x_t^R \leq 115$$
$$s_t^R \leq 55$$
$$x_t^S \leq 110$$
$$s_t^S \leq 60$$
$$x_t^R, s_t^R, x_t^S, s_t^S \geq 0$$
$$x_t^R, x_t^S, s_t^R, s_t^S \epsilon \mathbb{R}, \forall t \in \{1 \dots 5\}$$

Proof of Existence of Solution

1.) Based on the problem formulation the feasible set of solutions ($y_t^R,\ x_t^R,\ s_t^R, y_t^S,\ x_t^S,\ s_t^S$) to the BILP are bounded below by 0 to prevent ordering and holding negative units and bounded above to mimic storage space and production rate constraint. Thus, since a ball of radius ε = 116 can be created around the feasible set, the set is bounded.
2.) There exists a feasible solution to the problem as shown in Table 2. The solver will give this option should it not find another solution that further lowers the cost function.

t	x^R	x^S	s^R	s^S	y^R	y^S	**Retailer Cost**	**Supplier Cost**
1	79	89	25	10	1	1	1830	1208
2	110	100	32	0	1	1	2506	1230
3	110	110	26	0	1	1	2458	1350
4	110	110	31	0	1	1	2498	1350
5	110	110	0	0	1	1	2250	1350
						Individual Total	**$11,542.00**	**$6,488.00**
						Combined Total	**$18,030.00**	

Table 2: One feasible Solution to the Minimization Problem

FONC & SONC

For a minimum to exist it must satisfy the First Order Necessary and Second Order Necessary conditions.[x] The program used to solve for the optimal solution reduces the BILP to an LP (general form shown below). In order to determine the FONC and SONC conditions we defined $f(x)$ as the objective function (1) and take the gradient and Hessian, respectively. As can be seen from (2) the First Order Necessary conditions are never satisfied for an LP on the interior of the solution set since the gradient of $f(x)$ is a constant. Therefore in order for a solution to exist the optimal solution must be on the boundary. The SONC (3) is always satisfied and does not give any information about the minimum.

$$\begin{aligned} (\text{LP}) = \quad \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \quad A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m \\ & Bx = d, \quad B \in \mathbb{R}^{p\times n}, d \in \mathbb{R}^p \\ & x \in \mathbb{R}^n \end{aligned}$$

$$f(x) \equiv c^T x \quad (1)$$

$$\nabla f(x^*) = c^T \neq 0 \quad (2)$$

$$\nabla^2 f(x^*) = 0 \leq 0 \quad (3)$$

The KKT Conditions

$$x^* \; regular \qquad (1)$$

$$x^* \; feasible: g(x^*) \leq 0, \;\; h(x^*) = 0 \qquad (2)$$

$$\nabla_x f(x^*) + [Dg(x)]^T \mu^* + [Dh(x^*)]^T \lambda^* = 0 \qquad (3)$$

$$\mu^{*T} g(x^*) = 0, \mu^* \geq 0 \qquad (4)$$

Since the number of equations and constraint grows with the stop time T we account for the KKT condition assuming a stop time of T =1. The following KKT conditions are shown below. Since y_t is treated like a parameter we define $x^*, f(x), h(x^*),$ and $g(x^*)$.

$$x = [x_1^R \quad s_1^R \quad x_2^R \quad s_2^R]$$

$$f(x) = 20x_1^R + 8s_1^R + 12x_1^S + 11x_1^S$$

$$\nabla f(x^*) = \begin{bmatrix} 20 \\ 8 \\ 12 \\ 11 \end{bmatrix}$$

$$h(x^*) = \begin{bmatrix} x_1^R - s_1^R - 54 \\ x_1^S - x_1^R - s_1^S \end{bmatrix}$$

$$D\big(h(x^*)\big) = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & -1 \end{bmatrix}$$

$$g(x^*) = \begin{bmatrix} x_1^R - 115 \\ s_1^R - 55 \\ x_1^S - 110 \\ s_1^s - 60 \end{bmatrix}$$

$$D\big(g(x^*)\big) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I \ \in R^{4x4}$$

From condition 3, 4 equations are generated:

$$20 + \lambda_1^* - \ \lambda_2^* + \mu_1^* = 0$$

$$8 - \lambda_1^* + \mu_2^* = 0$$

$$12 + \lambda_2^* + \mu_3^* = 0$$

$$11 - \lambda_2^* + \mu_4^* = 0$$

From condition 4, another 4 equations are generated:

$$\mu_1^*(x_1^R - 115) = 0$$

$$\mu_2^*(s_1^R - 55) = 0$$

$$\mu_3^*(x_1^S - 110) = 0$$

$$\mu_4^*(s_1^s - 60) = 0$$

The 8 equations above and the 2 equation from $h(x)$ give a total of 10 equations in 10 unknowns ($x_1^R, s_1^R, x_1^S, s_1^S, \lambda_1^*, \lambda_2^*, \mu_1^*, \mu_2^*, \mu_3^*, \mu_4^*$). No constraints in $g(x)$ were active for this problem setup since the ordering and holding amount of one actor affects its own ordering and holding costs in other months. The ordering and holding amounts of one actor also affects the other actor. Therefore there exists no reason as to why it is necessary for the retailer or supplier to order or hold the maximum allowable amount and thus there are no active constraints. The number of equations in unknowns grows as a function of $10T$. For $T = 5$ there will be 50 equations in 50 unknowns. Combined with the fact that the solver must evaluate $2^{\mathrm{T+1}}$ combinations makes this problem difficult.

After setting up and solving the problem it can be observed that both the supplier and retailer order the maximum possible amount for most of the months but not all. This means that the mu values associated with the ordering constraints are nonzero since the constraints are active. However, the mu values associated with the storage constraint will be zero because the storage maximum is never reach making the constraints inactive. The optimal solution was found to satisfy the condition 2. The point is feasible as it satisfies the inequality and equality constraints:

$$g(x^*) = \begin{bmatrix} x_1^R - 115 \\ s_1^R - 55 \\ x_1^S - 110 \\ s_1^S - 60 \end{bmatrix} = \begin{bmatrix} 79 - 115 \\ 25 - 55 \\ 79 - 110 \\ 0 - 60 \end{bmatrix} = \begin{bmatrix} -36 \\ -30 \\ -31 \\ -60 \end{bmatrix} \leq 0$$

$$h(x^*) = \begin{bmatrix} x_1^R - s_1^R - 54 \\ x_1^S - x_1^R - s_1^S \end{bmatrix} = \begin{bmatrix} 79 - 25 - 54 \\ 79 - 79 - 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Results

Optimization A for Retailer (5 Month Analysis):

Following the steps outlined in the problem description, the 5 month optimization setup for the retailer with substituted values is as follows.

$$\min \sum_{t=1}^{5} 50y_t^R + 20x_t^R + 8s_t^R$$

$$s_0^R + x_1^R = 54 + s_1^R \quad (1)$$
$$s_1^R + x_2^R = 103 + s_2^R \quad (2)$$
$$s_2^R + x_3^R = 116 + s_3^R \quad (3)$$
$$s_3^R + x_4^R = 105 + s_4^R \quad (4)$$
$$s_4^R + x_5^R = 141 + s_5^R \quad (5)$$
$$x_t^R \le 115$$
$$s_t^R \le 55$$
$$x_t^R, s_t^R \ge 0$$
$$s_0^R = 0$$
$$x_1^R, x_2^R, x_3^R, s_1^R, s_2^R, s_3^R \epsilon\ \mathbb{R}, \forall t \in \{1 \ldots 5\}$$

t	x^R	s^R	y^R	**Retailer Cost**
1	59	5	1	1270
2	115	17	1	2486
3	115	16	1	2478
4	115	26	1	2558
5	115	0	1	2350
			Individual Total	**$11,142.00**

Table 4: Optimal Replenishment Plan for Retailer

Based on the market demand the minimum cost the retailer faces under this solution is $11,142. Table 4 represents the optimal solution without knowledge of the supplier's constraints. The retailer thinks that this is the lowest cost it can achieve over this time span. In the asymmetric case, the supplier must now solve its own optimization problem given $x_1^R, x_2^R, x_3^R, x_4^R$, and x_5^R. It is common for the unit ordering amount of the supplier to be tighter than that of the retailer. If it were not, the supplier would not incur a storage cost and the problem would not be interesting.

Optimization A for Supplier (5 Month Analysis)

$$\min \quad \sum_{t=1}^{5} 30y_t^S + 12x_t^S + 11s_t^S$$

$$s_0^S + x_1^S = x_1^R + s_1^S \ (6)$$
$$s_1^S + x_2^S = x_2^R + s_2^S \ (7)$$
$$s_2^S + x_3^S = x_3^R + s_3^S \ (8)$$
$$s_3^S + x_4^S = x_4^R + s_4^S \ (9)$$
$$s_4^S + x_5^S = x_5^R + s_5^S \ (10)$$
$$x_t^S \leq 110$$
$$s_t^S \leq 60$$
$$x_t^S, s_t^S \geq 0$$
$$x_1^S, x_2^S, x_3^S, s_1^S, s_2^S, s_3^S \epsilon \ \mathbb{R}, \forall t \in \{1 \dots 5\}$$

t	x^S	s^S	y^S	**Supplier Cost**
1	79	25	1	1253
2	110	32	1	1702
3	110	26	1	1636
4	110	31	1	1691
5	110	0	1	1350
			Individual Total	**$7,632.00**

Table 5: Optimal Replenishment plan for Supplier

The optimal cost for the supplier is $7,632. This solution in Table 5 represents an upper bound to the costs the supplier faces under asymmetric information. After solving problem A twice for both parties with asymmetric information, the optimal solution for each company was found. It is now up to the supplier to solve the problem under complete information to generate a solution with a lower cost to himself and the retailer. To do this one must find the optimal solution under optimization B with information being symmetric for comparison.

Optimization B for both Retailer and Supplier

The optimization process below assumes that the retailer and supplier are cooperating and sharing information. The new cost function is just a linear combination of the previous two.

$$(BILP) = \min \sum_{t=1}^{5} 50y_t^R + 20x_t^R + 8s_t^R + 30y_t^S + 12x_t^S + 11s_t^S$$

$$\text{s.t.}$$

$$\begin{aligned}
s_0^R + x_1^R &= 54 + s_1^R \quad (1)\\
s_1^R + x_2^R &= 103 + s_2^R \quad (2)\\
s_2^R + x_3^R &= 116 + s_3^R \quad (3)\\
s_3^R + x_4^R &= 105 + s_4^R \quad (4)\\
s_4^R + x_5^R &= 141 + s_5^R \quad (5)\\
s_0^S + x_1^S &= x_1^R + s_1^S \quad (6)\\
s_1^S + x_2^S &= x_2^R + s_2^S \quad (7)\\
s_2^S + x_3^S &= x_3^R + s_3^S \quad (8)\\
s_3^S + x_4^S &= x_4^R + s_4^S \quad (9)\\
s_4^S + x_5^S &= x_5^R + s_5^S \quad (10)\\
s_0^R, s_0^S &= 0 \qquad (11)\\
x_t^R &\le 115\\
s_t^R &\le 55\\
x_t^S &\le 110\\
s_t^S &\le 60\\
x_t^R, s_t^R, x_t^S, s_t^S &\ge 0\\
x_t^R, x_t^S, s_t^R, s_t^S &\epsilon\ \mathbb{R}, \forall t \in \{1,2,3,4,5\}
\end{aligned}$$

The solution is as follows.

Coordination Case									
t	d	x^R	x^S	s^R	s^S	y^R	y^S	**Retailer Cost**	**Supplier Cost**
1	54	79	79	25	0	1	1	1830	978
2	103	110	110	32	0	1	1	2506	1350
3	116	110	110	26	0	1	1	2458	1350
4	105	110	110	31	0	1	1	2498	1350
5	141	110	110	0	0	1	1	2250	1350
							Individual Totals	**$11,542.00**	**$6,378.00**
							Combined Total	**$17,920.00**	

Table 6: Optimal Replenishment Plan

A comparison of the final costs for the two companies and total cost are given in Table 7.

Case	Retailer Cost	Supplier Cost	Total Cost
Separate	$11,142.00	$7,632.00	$18,774.00
Combined	$11,542.00	$6,378.00	$17,920.00

Table 7: Individual and Total Costs Under Asymmetric and Symmetric Information

The optimal total cost to the retailer and the supplier is $17,920.00. Note that the supplier's cost decreased while that of the retailers increased. This happened because the supplier convinced the retailer to put more in storage to decrease its own storage. However, the increase in the retailer's cost was not the same as the decrease in the supplier's cost. The retailer had to pay an extra $400 dollars while the supplier saved $1,254 dollars. Thus, the supplier can cover the $400 and still decrease its own costs by $854. This remaining amount could be split halfway between the companies, but the supplier can keep as much as the retailer will allow before not accepting the deal. Thus, cooperation decreases the overall cost for both parties.

Sensitivity Analysis

A sensitivity analysis for the upper bounds on ordering and storage is given in Table 8. The upper bound on ordering for the supplier and retailer were the only constraints that showed variation.

The supplier ordering constraint has a large effect on cost as supplier orders are at a maximum after month 2 in the original solution, so decreasing the upper bound will force the supplier to store more units in month 1. The retailer ordering constraint only causes an increase in the objective function when the limit is 105 units. This is because the retailer will need to store more units in the prior month to meet demand. At a cap of 105 units, extra storage must be made so cost increases, but at a cap of 100 units, the feasible set is empty and there exists no solution under the set of constraints. Storage constraints did not affect the cost as storage was never near its upper bound as minimizing cost also minimizes storage cost.

Table of Upper Bound and Costs							
X^S	$f(x)$	X^R	$f(x)$	S^S	$f(x)$	S^R	$f(x)$
95	--	100	--	45	17920	40	17920
100	--	105	18320	50	17920	45	17920
105	18320	110	17920	55	17920	50	17920
110	**17920**	**115**	**17920**	**50**	**17920**	**55**	**17920**
115	17520	120	17920	65	17920	60	17920
120	17520	125	17920	70	17920	65	17920
125	17520	130	17920	75	17920	70	17920

Table 8: Sensitivity Analysis on Variable Upper Bounds

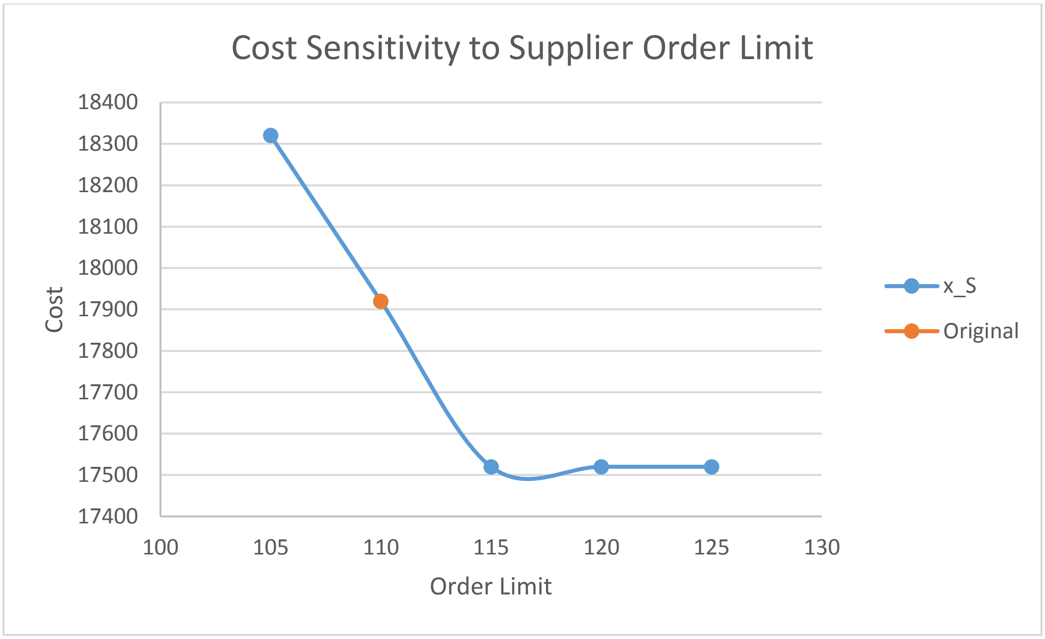

Figure 1: Sensitivity Analysis of Combined Cost with d_1

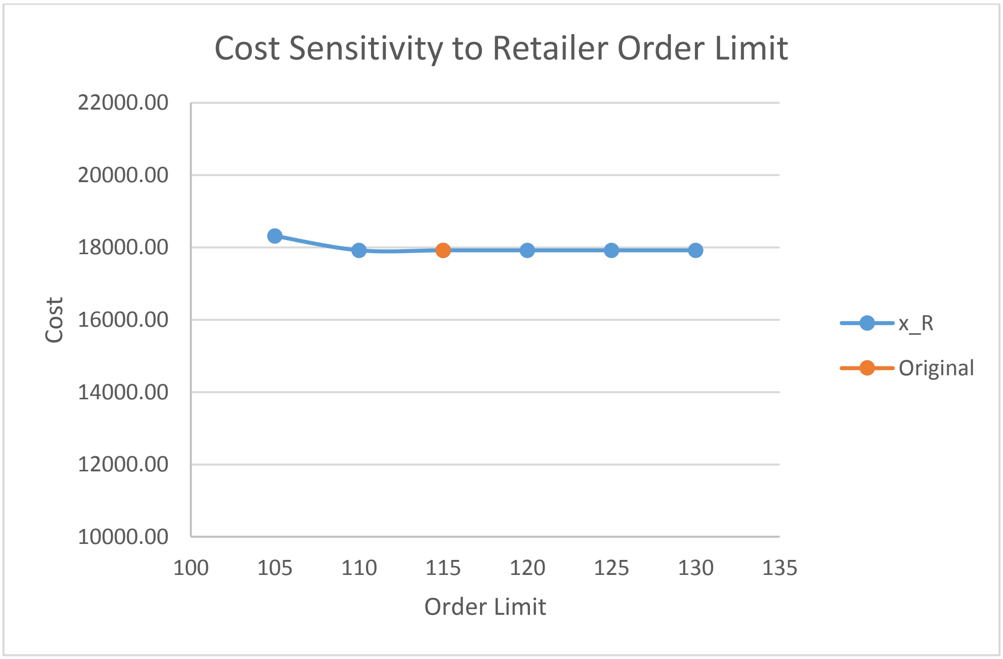

Figure 2: Sensitivity Analysis of Combined Cost with d_2

Shadow prices from the dual of the problem were obtained through Pulp for the combined case problem. The values are given in Table 9.

Observing the values for constraints, there are no constraints that are not sensitive to prices. There are however constraints that are more sensitive, as they have higher prices, and some with lower shadow prices indicating that they are less important. The most important constraints are constraints 5 and 9. Constraints 5 and 9 are balances on the units of shirts for month 4. These constraints may be sensitive due the high demand in month 5 of 141, which requires storage as the supplier can produce at most 110 units per month.

Shadow Price	**Value**
λ_1	32
λ_2	12
λ_3	40
λ_4	20
λ_5	48
λ_6	28
λ_7	56
λ_8	36
λ_9	64
λ_{10}	44

Table 9: Shadow Prices for Constraints 1-10 in Optimization Problem B

A sensitivity analysis was also performed on the demands for each month, where only one month's demand was varied and the objective function recorded. Table 10 gives the values of the cost for variations in $d_1, d_2, d_3, d_4,$ and d_5. There is clear increase in cost as the demand increases for each month as one would expect.

Demand Sensitivity Analysis									
$\boldsymbol{d_1}$	$\boldsymbol{f(x)}$	$\boldsymbol{d_2}$	$\boldsymbol{f(x)}$	$\boldsymbol{d_3}$	$\boldsymbol{f(x)}$	$\boldsymbol{d_4}$	$\boldsymbol{f(x)}$	$\boldsymbol{d_5}$	$\boldsymbol{f(x)}$
39	$17,440	88	$17,320	101	$17,200	90	$17,080	126	$16,960
44	$17,600	93	$17,520	106	$17,440	95	$17,360	131	$17,280
49	$17,760	98	$17,720	111	$17,680	100	$17,640	136	$17,600
54	**$17,920**	**103**	**$17,920**	**116**	**$17,920**	**105**	**$17,920**	**141**	**$17,920**
59	$18,080	108	$18,120	121	$18,160	110	$18,200	146	$18,240
64	$18,240	113	$18,320	126	$18,400	115	$18,480	151	$18,560
69	$18,400	118	$18,520	131	$18,640	120	$18,760	156	$18,880

Table 10: Sensitivity Analysis of Cost based on Demand over 5 Month Period

The changes in the cost due to increases in demand parameters are all linear for the problem. The change in cost per unit change in demand is given for each month in Table 11. The slopes also have a linear increase. The first month's increase in costs will simply be the sum of unit ordering costs for the retailer and supplier which $32. Increases in demand in the second month will require extra storage in month 1 as the ordering amount for the retailer in month 2 is already at the maximum. This is also true for months 3, 4, and 5. Thus an increase in each of these months will result in a ordering cost of $32 and additional storage cost for the retailer $8 per month per shirt ordered. Thus the additional unit storage costs for months 2, 3, 4 and 5 will be 8, 16, 24, and 32, respectively.

Parameter	**Slope**
d_1	32
d_2	40
d_3	48
d_4	56
d_5	64

Table 11: Change in Objective Function for Change in Parameter (Slope)

Conclusion

This report solved for the optimal replenishment strategy under symmetric and asymmetric information to minimize costs. The results of the combined case show that the two companies indeed benefit from cooperation and symmetric information sharing with an optimal value of $17,920 as compared to the asymmetric solution, which gives an optimal value of $18,774. The retailer's decision to share information drives down the combined cost, since the supplier could avoid unnecessary storage costs. This benefit rests on the assumption that the supplier would make side payments to the retailer in order to ensure that both costs are driven down.

Sensitivity analysis to demand shows that cooperation will involve one party increasing its own cost if the overall cost decreases. As demand increases the storage requirements of the retailer increase. The retailer chooses to store more product due to lower storage costs than the supplier. The supplier in turn avoids a larger increase in cost due to storage and can compensate the retailer, allowing both parties to avoid large increases in cost.

The decrease in cost over 5 months is $854. For a larger company, with a production scaled up by a factor of 100, demands will increase from 54, 103, 116, 105, and 141 to 5400, 10300, 11600, 10500, and 14100, which will in turn (assuming other limits also scale up) have a decrease in cost of $85,400 from information sharing, which is a significant amount. The difference in the cost between the symmetric and asymmetric cases will also increase with a longer time period of analysis, making it more significant to the companies. It is in the case of a large volume of product and long time periods of business that such symmetric information sharing will bring the most benefit to both companies. However, it should be noted that increasing the time period over which to generate a replenish plan will increase the number of LPs that need to be solved and will increase the run time of the program.

\

Future Work

Future work could be performed to make a more realistic model by removing simplifications, but also by generalizing the problem further. One way to generalize this problem would be to account for perishable goods, which are goods that lose value the longer they are in storage such as food. The longer they are in storage the larger the discount the seller will need to give in order to make a sale. This would lead to an adaptation to the storage cost term in the objective function that would increase costs over time and could thus affect the optimal solution. Shipping prices and product prices could also be modeled over time instead of being assumed to be constant. These additions can potentially make the objective function into an MINLP.

One possible way to better model the problem would be to change the order pricing parameter for the retailer and the supplier. Shipping costs are modeled in this paper using a single cost for making an order. Shipping costs will only change depending on whether or not an order is made. The model could be improved by specifying the shipping cost of shirts of a certain batch size (e.g. 100 shirts) and an associated shipping cost. This would tie the shipping cost to the number of units ordered as well as the choice to order shirts.

Another potential issue is the ordering and selling of multiple products, where the supplier has a finite number of workers to produce any of the multiple products. In this situation, the production of one product is tied to another and adds more constraints to the problem. Other generalizations could include the supplier being a retailer itself. Now the supplier has to determine how much product to keep and sell itself to customers and to other retailers. These additions will make the problem more complex since there are now more cases that need to be looked at before an optimal replenishment plan can be found.

References

[i] Clothing store. Man and woman clothes shop and boutique. Shopping,.. https://www.123rf.com/photo_41645517_stock-vector-clothing-store-man-and-woman-clothes-shop-and-boutique-shopping-fashion-bags-accessories-flat-style-.html (accessed Dec 19, 2018).
[ii] Investopedia. https://www.investopedia.com/terms/e/economicorderquantity.asp (accessed November 11, 2018).
[iii] Phouratsamay, S.; Sidhoum, S; Pascual, F. Coordination of a two-level supply chain with contracts Optimization. [Online] **2018**, http://www.optimization-online.org/DB_FILE/2018/07/6702.pdf (accessed October 14, 2018).
[iv] Treasury Today. Introduction to Financial Supply Chain Management. http://treasurytoday.com/2007/01/introduction-to-financial-supply-chain-management (accessed November 11, 2018).
[v] Phouratsamay, S.; Sidhoum, S; Pascual, F. Coordination of a two-level supply chain with contracts under complete or asymmetric information. Optimization. [Online] **2017**, http://www.optimization-online.org/DB_FILE/2017/07/6123.pdf (accessed October 14, 2018).
[vi] PuLP documentation. https://pythonhosted.org/PuLP/ (accessed Dec 19, 2018).
[vii] Kalaoglu, Özlem. Et al. RETAIL DEMAND FORECASTING IN CLOTHING INDUSTRY. http://dergipark.gov.tr/download/article-file/218295. (accesed December 14, 2018)
[viii] JCPenney. https://www.jcpenney.com (accessed December 18, 2018).
[ix] FedEx. https://www.fedex.com/apps/shipping-rates#/ (accessed December 18, 2018).
[x] Chong, E.; Zak, S. An Introduction to Optimization, 4th ed.; John Wiley & Sons: Hoboken, NJ.

Appendix: Code for Retailer, Supplier, and Combined Case

```
"""
Ryan Jaipersaud, Armaan Thapar
ChE488, 12/19/2018

Combined Case of Retailer and Producer for 3 months under Asymmetric and Symmetric Information Exchange.

There are 3 functions that take in a demand vector and return the objective function value. The functions are currently already called
to take the demand vector based on collected data for 5 months. The orders from the retailer are the demand for the supplier, and these are directly
sent to the supplier in the code, using an extra parameter in the retailer function. The program currently uses 5 months of data (can take more- will
adjust parameters automatically) as calculations for sensitivity analysis would take too long to calculate using 6 months or more. """

import numpy as np
from pulp import *
import itertools
import math
import pandas as pd

# Demand for each month
d = [54, 103, 116, 105, 141]

# Retailer Solution (Asymmetric Information Exchange)
def Retailer_Case(demand, supplier_demand=False): # supplier_demand = True will print orders as a list for supplier orders

    # Use demand to create indices and y values for brute force solution finding
    d = demand # Set d to the demand vector given for the number of months
    n = len(d) # number of combinations of ordering plans
    lst = list(map(list,itertools.product([0, 1], repeat=n))) # maps a combination of orders to a list of all combinations of orders
    N = len(d) # Number of months
    indices = [i for i in range(1,N+1)] # List of indices representing months (Used to create variables)

    # Variables
    x_r = LpVariable.dicts("xR",indices, lowBound=0, upBound=115) # Create N variables as a list under x_r (xR_1,..., xR_5)
    s_r = LpVariable.dicts("sR",indices, lowBound=0, upBound=55)  # Create N variables as a list under s_r (sR_1,..., sR_5)
```

```
objective_func = math.inf # Intitialize the minimum objective function to infinity (avoid ignoring large solutions)

for i in range(len(lst)):
    y_r = lst[i]    # Take one combination of values for the y vector

    prob = LpProblem("Supply Chain Problem: Retailer Case", LpMinimize) # Initialize problem in PuLP, specify that we are minimizing

    term1  = 0  # Want to make single term for objective function, as the second input is automatically taken to be a constraint by PuLP
    for i in range(0,len(indices)):
        term1 += 50*y_r[i]+20*x_r[i+1]+8*s_r[i+1]
    prob += term1, "Objective Function" # Add objective function

    # Add iteratively using the Constraint equations from problem
    for i in range(N):
        if i == 0:
            prob += x_r[i+1]*y_r[i] == d[i] + s_r[i+1], "Constraint " + str(i+1) # Constraint for month 1 (assume no initial storage)
        else:
            prob += s_r[i] + x_r[i+1]*y_r[i] == d[i] + s_r[i+1], "Constraint " + str(i+1)  # Constraints for retailer

    prob.solve()  # Solve Problem
    if LpStatus[prob.status] == "Optimal":    # Checks if point is feasible (basic feasible solution)

        if value(prob.objective) <  objective_func:   # Checks if value is lower than previous
            objective_func = value(prob.objective)    # Stores value if it is lower
            lowest_soln = np.array([])                # Initialize array to store variables from minimum
            for v in prob.variables():
                lowest_soln = np.append(lowest_soln, v.varValue)    # Store variables into array

            y_min = y_r                               # Stores y vector at solution as y_r

            data = {'x_R': pd.Series(lowest_soln[N:2*N+1], index = indices), # Store data as dictionary
                    'S_R': pd.Series(lowest_soln[0:N], index = indices),
                    'y_R': pd.Series(y_min, index = indices)}

            # Create and format dataframe to show t as title for index
            df = pd.DataFrame(data)
            df.index.name = ''
            df = df.rename_axis('').rename_axis("t", axis="columns")

    # Prints orders as a vector of demands for the supplier
    print("Retailer Case")
    print("Objective Function:",value(prob.objective))
    print(df)

    # Creates a vector of x_R for Supplier inputs
    if supplier_demand:
        demands = []
        for v in prob.variables()[N:2*N]:  # indices 0 to 2 are storage, 3 to 6 are orders
            demands.append(v.varValue)
        return demands
    else:
        return objective_func

# Supplier Solutions (Asymmetric Information Exchange)
def Supplier_Case(demand):

    # Use demand to create indices and y values for brute force solution finding
    d = demand # Set d to the demand vector given for the number of months
    n = len(d) # number of combinations of ordering plans
    lst = list(map(list,itertools.product([0, 1], repeat=n))) # maps a combination of orders to a list of all combinations of orders
    N = len(d) # Number of months
    indices = [i for i in range(1,N+1)] # List of indices representing months (Used to create variables)

    # Variables for Supplier
    x_s = LpVariable.dicts("xS",indices, lowBound=0, upBound=110) # Create N variables as a list under x_s (xS_1,..., xS_5)
    s_s = LpVariable.dicts("sS",indices, lowBound=0, upBound=60)  # Create N variables as a list under s_s (sS_1,..., xS_5)

    objective_func = math.inf # Intitialize the minimum objective function to infinity (avoid ignoring large solutions)

    for i in range(len(lst)):
        y_s = lst[i]       # Take one combination of values for the y vector
    prob = LpProblem("Supply Chain Problem: Supplier Case", LpMinimize) # Initialize problem in PuLP, specify that we are minimizing

    # Objective Function
    term1  = 0                    # Add as single term as for retailer
    for i in range(len(indices)):
        term1 += 30*y_s[i]+12*x_s[i+1]+11*s_s[i+1]
    prob += term1, "Objective Function"

    # Add constraints
    for i in range(N):
        if i == 0:
            prob += x_s[i+1]*y_s[i] == d[i] + s_s[i+1], "Constraint " + str(i+1)    # Month 1 constraint (assumes 0 initial storage)
        else:
            prob += s_s[i] + x_s[i+1]*y_s[i] == d[i] + s_s[i+1], "Constraint " + str(i+1)  # Months 2 onwards constraints

    prob.solve()   #Solves problem
    if LpStatus[prob.status] == "Optimal":     # Checks if point is feasible (basic feasible solution)

        if value(prob.objective) <  objective_func:    # Check if value is lower than previous
            objective_func = value(prob.objective)     # Store value if it is lower
            lowest_soln = np.array([])                 # Initialize array to store variables from minimum
            for v in prob.variables():
                lowest_soln = np.append(lowest_soln, v.varValue)   # Store variables into array

            y_min = y_s                                      # Store y vector at minimum

            data = {'x_S': pd.Series(lowest_soln[N:2*N+1], index = indices),  # Store data as dictionary
                    's_S': pd.Series(lowest_soln[0:N], index = indices),
                    'y_S': pd.Series(y_min, index = indices)}

            df = pd.DataFrame(data)  # Create dataframe
            df.index.name = ''
            df = df.rename_axis('').rename_axis("t", axis="columns")  # Add t as title for index


# Print objective function and dataframe of results
```

```
    print("\nSupplier Case")
    print("Objective Function:",value(prob.objective))
    print(df)

    return objective_func



# Combined Case Solution (Symmetric Information Exchange)
def Combined_Case(demand, order_r_ub = 115, order_s_ub = 110, store_r_ub = 55, store_s_ub = 60): # Kept optional parameters for testing
sensitivity

    # Use demand to create indices and y values for brute force solution finding
    d = demand # Set d to the demand vector given for the number of months
    n = len(d)*2 # number of combinations of ordering plans
    lst = list(map(list,itertools.product([0, 1], repeat=n))) # maps a combination of orders to a list of all combinations of orders
    N = len(d) # Number of months
    indices = [i for i in range(1,N+1)] # List of indices representing months (Used to create variables)

    # Retailer
    x_r = LpVariable.dicts("xR",indices, lowBound=0, upBound=115) # Create N variables as a list under x_r (xR_1,..., xR_5)
    s_r = LpVariable.dicts("sR",indices, lowBound=0, upBound=55)  # Create N variables as a list under s_r (sR_1,..., sR_5)

    objective_func = math.inf # Intitialize the minimum objective function to infinity (avoid ignoring large solutions)

    # Supplier
    x_s = LpVariable.dicts("xS",indices, lowBound=0, upBound=110) # Create N variables as a list under x_s (xS_1,..., xS_5)
    s_s = LpVariable.dicts("sS",indices, lowBound=0, upBound=60)  # Create N variables as a list under s_s (sS_1,..., xS_5)

    for i in range(len(lst)): # iterates over all possible order combinations for y
        y = lst[i] # assigns combination i from lst
        y_r = y[0:N] # takes the first n/2 from combination i and assigns to it retailer ordering plan
        y_s = y[N:int(2*N)] # takes the first n/2 from combination i and assigns to it supplier ordering plan

        prob = LpProblem("Combined Case Problem: Combined Case", LpMinimize) # Initialize problem in PuLP, specify that we are minimizing
```

```
# Objective Function (once again combine into one term to avoid parts being taken as constraints by PuLP)
term1  = 0
term2  = 0
for i in range(0,N):#len(indices)+1):
    term1 += 50*y_r[i]+20*x_r[i+1]+8*s_r[i+1]
    term2 += 30*y_s[i]+12*x_s[i+1]+11*s_s[i+1]
prob += term1 + term2, "Objective Function" # Assign the objective function

# Constraints
for i in range(N):

    if i == 0:
        # Constraints for retailer and supplier for month 1
        prob += x_r[i+1]*y_r[i] == d[i] + s_r[i+1], "Retailer Constraint " + str(i+1)
        prob += x_s[i+1]*y_s[i] == x_r[i+1] + s_s[i+1], "Supplier Constraint " + str(i+1)
    else:
        # Constraints for month 2 onwards
        prob += s_r[i] + x_r[i+1]*y_r[i] == d[i] + s_r[i+1], "Retailer Constraint " + str(i+1)
        prob += s_s[i] + x_s[i+1]*y_s[i] ==  x_r[i+1]+ s_s[i+1], "Supplier Constraint " + str(i+1)

prob.solve() # solve the problem under the constraints

if LpStatus[prob.status] == "Optimal": # Only accept optimal solutions
    if value(prob.objective) <  objective_func: # Only accept solutions that lower the objective function

        objective_func = value(prob.objective) # Define value for objective function to print later on

        lowest_soln = np.array([])  # Initialize array to store variables
        for v in prob.variables():
            lowest_soln = np.append(lowest_soln, v.varValue) # Store variables

        y_min_r = y_r # Store values for y for both retailer and supplier
        y_min_s = y_s

        data = {'x_R': pd.Series(lowest_soln[2*N:3*N], index = indices),   # Store data as dictionary
              'x_S': pd.Series(lowest_soln[3*N:4*N], index = indices),
                  's_R': pd.Series(lowest_soln[0:N], index = indices),
                  's_S': pd.Series(lowest_soln[N:2*N], index = indices),
                   'y_R': pd.Series(y_r, index= indices),
                   'y_S': pd.Series(y_s, index= indices)}

            df = pd.DataFrame(data) # conver data to dataframe
            df.index.name = ''
            df= df.rename_axis('').rename_axis("t", axis="columns")



            main_prob = prob
    # Print results
    print("\nCombined Case")
    print('Objective Function:',objective_func)
    print(df)

    return objective_func

Supplier_Case(Retailer_Case(d, supplier_demand=True)) # Will print results for both retailer and supplier
Combined_Case(d)                                      # Will print results for combined case
```