Ryan Jaipersaud
Armaan Thapar
First Draft

Professor Davis
November 12, 2018
Convex Optimization

Abstract

It is in the interest of any company that sells a product to minimize the cost of shipping, ordering and storage costs. One factor in reducing costs is to find an optimal replenishment strategy, while meeting market demand to maximize profit. An optimal replenishment strategy can determine whether a company makes a small or large profit. This paper explores the difference between a retailer acting independently versus cooperating with its supplier in a two-chain supply management problem. The diagram below 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. In this paper, a startup retailer and supplier selling shirts are negotiating their optimal replenishment strategy. The retailer pays $10 per shipment, $5 per shirt, and $4 per shirt in storage. The supplier pays $7 per shipment, $3 per shirt, and $11 per shirt in storage. The market demand was for 3 months and is as follows: 10, 50, and 90 shirts. The ordering/storage upper limit constraints for the retailer and supplier are 60/30 shirts and 55/10 shirts, respectively. Acting independently, the total cost for the retailer and supplier is $980 and $636, respectively. However, if the two companies work together the costs are $1020 and $526, respectively. The results are summarized in Table 1. The supplier then offers a side payment of $40 dollars to cover the increase in the retailers cost while decreasing its own by $70 overall.

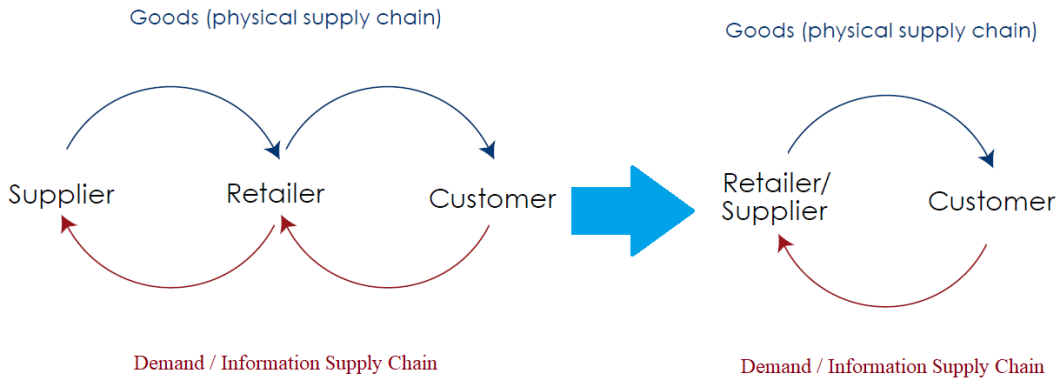

Diagram 1: Flow of Goods and Information in a Supply Chain[1]

Case	Retailer Cost	Supplier Cost	Total Cost
Asymmetric	$980.00	$636.00	$1,616.00
Symmetric	$1,020.00	$526.00	$1,546.00

Table 1: Individual and Total Costs Under Asymmetric and Symmetric Information

[1] Treasury Today. Introduction to Financial Supply Chain Management. http://treasurytoday.com/2007/01/introduction-to-financial-supply-chain-management (accessed November 11, 2018).

Introduction

In supply chain management, finding the optimal replenishment plan, which is the number of units to buy and hold over the course of a time period, proves to be a difficult problem. The cost of a product is tied up in 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.[2] 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 sell 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 their cost by attempting to meet demand exactly. In this paper 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.

Formalized in scalar form below, all companies aim to minimize the cost function $C(x, y, s)$. There are there $3T + 2$ decision variables ($x_t, y_t, s_t \quad \forall t \in \{1,2,3\}$ as well as s_0^R and s_0^S) which grow as a function of T, which affects the complexity of the program. It is the company's job to determine all $3T + 2$ variables to minimize the cost. Equation 1 ensures demand is met. Equation 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. Equation 3 is a positivity constraint. Equation 4 is a limit on the amount of storage. Note that only d, S, X and T are parameters.

Optimization Problem A

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

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

$$x_t \leq X \leq d_{tT} y_t \quad \forall t \in \tau, \quad (2)$$

$$x_t, s_t \geq 0 \quad \forall t \in \tau, \quad (3)$$

$$s_t \leq S \quad \forall t \in \tau, (4)$$

$$y_t \in \{0,1\}$$

$$x_t, s_t \in \mathbb{R}$$

[2] Investopedia. https://www.investopedia.com/terms/e/economicorderquantity.asp (accessed November 11, 2018).

Definition of variables

$$t = \text{time index}$$
$$T = \text{stop time}$$
$$x_t = \text{units to be ordered}$$
$$y_t = \text{placing an order}$$
$$s_t = \text{units in storage}$$
$$d_{tT} = \text{cumulative demand from t to T} = \sum_{i=t}^{T} d_i$$
$$d_i = \text{market demand at time i}$$

The goal of this study is to analyze a two-level supply chain minimization problem. In this problem, there exist two actors, a retailer who is responsible for selling goods and a supplier who is responsible for making the goods. Individually both companies are running their own optimization algorithm to solve optimization problem A to minimize their cost. 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.[3] However, the replenishment strategy forced by the retailer can prove unfavorable to the supplier. 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.[4]

Optimization Problem B

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

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

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

$$x_t^R \leq X^R \leq \ d_{tT} y_t^R \ \ \forall t \in \tau, \qquad (3)$$

$$x_t^S \leq \ X^S \leq \ x_{tT}^R y_t^S \ \ \forall t \in \tau, \qquad (4)$$

$$s_t^R \leq \ S^R \ \ \forall t \in \tau, \qquad (5)$$

$$s_t^S \leq \ S^S \ \forall t \in \tau, \qquad (6)$$

[3] 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).
[4] 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).

$$x_t^R, s_t^R, x_t^S, s_t^S \geq 0 \ \ \forall t \in \tau, \quad (7)$$

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

$$y_t^R, y_t^S \epsilon \{0,1\}; \ x_t^R, x_t^S, s_t^R, s_t^S \epsilon \mathbb{R}$$

In optimization problem B superscripts denote the company to which variables are assigned. In this case there are $(2(3T) + 2)$ or $(6T + 2)$ variables to be minimized over. For equations 2 and 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. If the cost the retailer faces after solving problem B is lower than the cost after solving 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:

min Ordering and Storage costs for the supplier and the retailer

s.t.
1.) The retailer meets market demand
2.) The supplier meets the retailer′s demand
3.) The retailer cannot order more than the order limit
4.) The supplier cannot order more than the order limit
5.) The retailer cannot surpass the storage limit
6.) The supplier cannot not surpass the storage limit
7.) The amount of units ordered and stored is greater than or equal to 0

Problem Description

A new startup retailer is working with a new startup t-shirt company 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. A client has notified the retailer that they will need 10, 50 and 90 shirts by the first of January, February, and March respectively. The supplier notifies the retailer that they can only produce 60 shirts a month and has an ordering cap of 55 shirts a month from the company prior in the chain. The retailer and the supplier can only hold up to 30 and 10 shirts a month in storage, respectively. The supplier buys shirts for $3 per shirt and sells them at $5 per shirt. The inventory cost for the retailer and the supplier are $4 and $11 per shirt, respectively. The shipping cost for the retailer and the supplier is $10 and $7 per order, respectively. The objective is to find the best replenishment plan that minimizes the cost to both actors.

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 scipy.optimize.linprog()
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 artificially created to model a retailer selling shirts and a supplier producing shirts over 3 months. The market demand was assumed to be 10, 50, and 90 shirts. The orders/storage upper limit constraints for the retailer and supplier are 60/30 shirts and 55/10 shirts, respectively. The shipping, ordering, and storage cost for retailer were $10 per shipment, $5 per unit ordered, $4 per unit in storage. The same costs for the supplier were and $7 per order, $3 per unit ordered, $11 per unit in storage, respectively. Approximate values for cost of shipping and cost per shirt were taken from the Allied Shirts website.[5]

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 thus approach negative infinity since omega would be unbounded.

The upper constraints on $s_t^R \leq 30$, $s_t^S \leq 10$ are to model real-world storage constraints. Storage units have a cap on how much one is allowed to store. The upper constraints on $x_t^R \leq 60$, $x_t^S \leq 55$ 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.

[5] Allied Shirts. https://www.alliedshirts.com/designer?cppid=133&quantity=1&colorId=260 (accessed November 11, 2018).

For now, y_t = 1 for all t for both retailer and supplier. This is a good assumption because based on the demand and the ordering limits for both the retailer and the supplier an order must occur at each t for a feasible solution to exist. This is set to make the programming easier later on. Otherwise the solver would set y_t to zero and yet still make an order for units.

By assuming the shipping costs, ordering cost per unit and storage cost per unit are constant over time for the retailer and supplier, the objective function and constraints reduce this to a linear programming problem with a convex function over a convex set (see Appendix for proof). There are many ways to solve an LP such as with a simplex, ellipsoid or an interior point method. This problem was solved by setting up optimization B in simplified form (see computational setup) and using the linear solver in scipy.optimize.linprog() to find the optimal replenishment plan. In this problem the method linprog() used was the simplex method. Based on the scaling of the problem the solution is sufficient to model two small companies.

Proof of Existence of Solution:

Based on the setup of the using the parameters above the objective function reduces to:

$$P = \min \sum_{t=1}^{T} 10y_t^R + 5x_t^R + 4s_t^R + \sum_{t=1}^{T} 7y_t^S + 3x_t^S + 11s_t^S$$

1.) Assuming $y_t = 1$, P is a polynomial function or order 1 and is thus continuous. Because P is of order one a line drawn between any two points on P will be still be on P and therefore P is convex.
2.) For every y_t, x_t and s_t, P is defined
3.) Assuming T = 3, P is a functional that maps R^{12} onto R

$$\text{s.t.} \quad \begin{gathered} x_t^R \leq 60 \\ s_t^R \leq 30 \\ x_t^S \leq 55 \\ s_t^S \leq 10 \\ x_t^R, s_t^R, x_t^S, s_t^S \geq 0 \\ y_t^R, y_t^S \epsilon \{1\};\ x_t^R, x_t^S, s_t^R, s_t^S \epsilon \mathbb{R} \end{gathered}$$

4.) Given the constraint above, omega includes its boundary and is thus closed. Because omega takes on the form $Ax \leq b$, (a collection of half spaces) the feasible set is convex.
5.) A ball of radius $\varepsilon = 61$, can be created around omega and thus omega is bounded

$$\begin{gathered} s_0^R + x_1^R = 10 + s_1^R \\ s_1^R + x_2^R = 50 + s_2^R \\ s_2^R + x_3^R = 90 + s_3^R \\ s_0^S + x_1^S = x_1^R + s_1^S \\ s_1^S + x_2^S = x_2^R + s_2^S \\ s_2^S + x_3^S = x_3^R + s_3^S \\ y_t^R, y_t^S = 1 \\ s_0^R, s_0^S = 0 \end{gathered}$$

6.) Given the set of equation above a potential solution exists and is shown below and therefore the feasible set of solution is non-empty.

Coordination Case									
Time	**Demand**	y^R	y^S	x^R	x^S	s^R	s^S	**Cost to Retailer**	**Cost to Supplier**
1	10	1	1	40	45	30	5	330	197
2	50	1	1	50	50	30	5	380	212
3	90	1	1	60	55	0	0	310	172
							Totals:	**$1,020.00**	**$581.00**

Table 2: One feasible Solution to the Minimization Problem

Since all criteria for Weierstrass's Theorem have been met, there exists a solution x^* which minimizes P.

FONC & SONC

For a minimum to exist it must satisfy the First Order Necessary and Second Order Necessary conditions.

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

Because the optimization setup above takes on the form of an LP, the FONC for an interior point is never satisfied and thus the optimal solution is assumed to exist on the boundary. The SONC is always satisfied and does not give any information about the minimum.

$$f(x) = c^T x$$

$$\nabla f(x^*) = c^T \neq 0 \; for \; all \; \; interior \; x \; \; (FONC)$$

$$\nabla^2 f(x^*) = 0 \leq 0 \; for \; all \; x \; (SONC)$$

Results

Optimization A for Retailer:

Following the steps outlined in the problem description, the optimization setup for the retailer with plugged in values is as follows. Computation setup is described in the next section.

$$\min \quad \sum_{t=1}^{T} C^R(x_t^R, y_t^R, s_t^R) \; = \; \sum_{t=1}^{T} 10y_t^R + 5x_t^R + 4s_t^R$$

$$\text{s.t.} \qquad x_t^R \leq 60$$
$$s_t^R \leq 30$$
$$x_t^R, s_t^R \geq 0$$

$$s_0^R + x_1^R = 10 + s_1^R$$
$$s_1^R + x_2^R = 50 + s_2^R$$
$$s_2^R + x_3^R = 90 + s_3^R$$
$$y_t^R = 1$$
$$s_0^R = 0$$
$$y_t^R \epsilon \{1\};\; x_t^R, s_t^R \epsilon \mathbb{R}$$

Time	Market Demand
1	10
2	50
3	90

Table 3: Market demand for shirts

Retailer Side					
Time	**Demand**	**y^R**	**x^R**	**s^R**	**Cost retailer**
1	10	1	30	20	240
2	50	1	60	30	430
3	90	1	60	0	310
				Total:	**$980.00**

Table 4: Optimal Replenishment Plan for Retailer

Based on the market demand the minimum cost the retailer faces under this solution is $980. 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. Now that the retailer knows what it wants, the supplier must solve their own optimization problem given x_t^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

$$\min \quad \sum_{t=1}^{T} C^S(x_t^S, y_t^S, s_t^S) \;=\; \sum_{t=1}^{T} 7y_t^S + 3x_t^S + 11s_t^S$$

$$\text{s. t.} \qquad x_t^S \le 55$$
$$s_t^S \le 10$$
$$x_t^S, s_t^S \ge 0$$

$$s_0^S + x_1^S = x_1^R + s_1^S$$
$$s_1^S + x_2^S = x_2^R + s_2^S$$
$$s_2^S + x_3^S = x_3^R + s_3^S$$
$$y_t^S = 1$$
$$s_0^S = 0$$
$$x_1^R = 30$$
$$x_2^R = 60$$
$$x_3^R = 60$$

$$y_t^S \epsilon \{1\};\; x_t^S, s_t^S \epsilon \mathbb{R}$$

Supplier Side					
Time	**Demand**	**y^S**	**x^S**	**s^S**	**Cost supplier**
1	30	1	40	10	237
2	60	1	55	5	227
3	60	1	55	0	172
				Total:	**$636.00**

Table 5: Optimal Replenishment plan for Supplier

The optimal cost for the supplier is $636. This solution in Table 5 represents an upper bound to the costs the supplier faces under asymmetric information. After solving problem A twice 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.

$$\min \quad \sum_{t=1}^{T} 10y_t^R + 5x_t^R + 4s_t^R + \sum_{t=1}^{T} 7y_t^S + 3x_t^S + 11s_t^S$$

$$\text{s.t.} \quad \begin{gathered} x_t^R \leq 60 \\ s_t^R \leq 30 \\ x_t^S \leq 55 \\ s_t^S \leq 10 \\ x_t^R, s_t^R, x_t^S, s_t^S \geq 0 \end{gathered}$$

$$\begin{gathered} s_0^R + x_1^R = 10 + s_1^R \\ s_1^R + x_2^R = 50 + s_2^R \\ s_2^R + x_3^R = 90 + s_3^R \\ s_0^S + x_1^S = x_1^R + s_1^S \\ s_1^S + x_2^S = x_2^R + s_2^S \\ s_2^S + x_3^S = x_3^R + s_3^S \\ y_t^R, y_t^S = 1 \\ s_0^R, s_0^S = 0 \end{gathered}$$

$$y_t^R, y_t^S \epsilon \{1\};\ x_t^R, x_t^S, s_t^R, s_t^S \epsilon \mathbb{R}$$

The solution is as follows.

Coordination Case									
Time	**Demand**	**y^R**	**y^S**	**x^R**	**x^S**	**s^R**	**s^S**	**Cost to Retailer**	**Cost to Supplier**
1	10	1	1	40	40	30	0	330	127
2	50	1	1	50	55	30	5	380	227
3	90	1	1	60	55	0	0	310	172
							Totals:	**$1,020.00**	**$526.00**

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	$980.00	$636.00	$1,616.00
Combined	$1,020.00	$526.00	$1,546.00

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

The optimal total cost to the retailer and the supplier is \$1546. 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 the first month ($s_t^R = 30$ versus 20 at t =1) to decrease its own storage (s_t^S = 0 versus 10 at t =1). 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 \$40 dollars while the supplier saved \$110 dollars. Thus, the supplier can cover the \$40 and still decrease while still decreasing its own costs by \$70. This shows that cooperation decreases the overall cost for both parties.

Computational Setup:

The original problems (optimization for retailer, supplier, and combined) can be posed as LPs in the form:

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

$Ax \leq b$ is the set of inequality constraints on variables contained in the vector x and $Bx = d$ is the set of equality constraints on the same set of variables in vector x. This form of the problem can be solved using the linprog function in the scipy.optimize library in python. The function scipy.optimize.linprog takes the arguments c, A, b, B, and d, which are analogous to the arguments in the LP along with other optional parameters for the method.[6] The function output gives the optimal value of the objective function as well as the vector x and other information about the obtained solution. In order to solve each problem, matrices and vectors are constructed based on the necessary variables. In the case of 3 months ($t = 1, 2, 3$) and the given demand schedule, 3 different problems are solved, and the results compared.

Retailer:

The first actor is the retailer designing an optimal replenishment schedule based on demand. This problem involves 10 variables ($x \in \mathbb{R}^{10}$). 3 values each for: x_t^R, y_t^R, and s_t^R plus one additional variable since s_0^R is needed in the first month. Matrix A of inequalities has 12 rows corresponding to constraints and 10 columns corresponding to the 10 variables ($A \in R^{12 \times 10}$). There are 6 rows for positivity constraints for each x_t^R and s_t^R, and another 6 rows to constrain the maximum values for x_t^R and s_t^R in each month. The vector b is set to match the correct product of matrix A with x to satisfy the inequality ($b \in \mathbb{R}^{12}$).

The matrix B is composed of 7 rows and 10 columns($B \in \mathbb{R}^{7 \times 10}$), with 10 columns for the 10 variables in x. There are 3 rows to satisfy the demand schedule ($s_{t-1}^R + x_t^R = d_t + s_t^R, \forall t = \{1,2,3\}$). There are an additional 3 rows for the simplification that all orders are accepted and only order quantity changes ($y_t^R = 1$). The last row set the initial storage to zero ($s_0^R = 0$) since analysis assumed zero initial products. Vector d is the set of corresponding results to the matrix equation $Bx = d$ ($d \in \mathbb{R}^7$). The created matrices and vectors can then be passed into linprog, which gives an optimum value of $980 for the retailer.

[6] scipy.optimize.linprog. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html (accessed Nov 13, 2018).

Supplier:

The problem of minimizing the cost to the supplier is very similar to that of the retailer. The difference is in the demand equation, as the demand for the supplier in each month will be the orders from the retailer, x_t^R. Thus, there are a total of 13 variables for the supplier ($x \in \mathbb{R}^{13}$), because of the additional 3 values of x_t^R, in addition to $x_t^S, y_t^S,$ and s_t^S.

The matrix A of inequality constraints for the supplier has 12 rows and 10 columns ($A \in \mathbb{R}^{12\times10}$). 6 rows constrain the maximum values for storage s_t^S and orders x_t^S. An additional 6 rows are positivity constraints on storage s_t^S and orders x_t^S. The vector b corresponds to the matrix-vector product of Ax ($b \in \mathbb{R}^{12}$).

The matrix B of equality constraints has 10 rows and 13 columns ($B \in \mathbb{R}^{10\times13}$). There are 3 rows to satisfy the demand schedule for the supplier given by: $s_{t-1}^S + x_t^S = x_t^R + s_t^S$. There are an additional 3 rows to set the values for $x_1^R, x_2^R,$ and x_3^R, which are the orders made to the supplier from the previous analysis. The last 3 rows set $y_t^R = 1$ for the order acceptance simplification. The last row sets the initial storage to zero since analysis assumes zero initial product. Vector d is composed of the corresponding results to the matrix equation $Bx = d$ ($d \in \mathbb{R}^{10}$). The new matrices and vectors can then be passed into linprog. This gives a result of \$636 for the cost to the supplier.

Combined Case:

The combined case of the retailer and supplier sharing information involves solving the problem simultaneously for the retailer and the supplier, and then reducing the total cost. This allows for individual costs to the retailer or supplier to increase if total cost decreases, with the assumption that side payments will be made by one of the companies to offset the other company's loss.

The combined case has a total of 20 variables ($x \in \mathbb{R}^{20}$). These are $x_t^R, x_t^S, y_t^R, y_t^S, s_t^R,$ and s_t^S for $t = \{1,2,3\}$ along with s_0^R, s_0^S. The matrix of inequality constraints A has 24 rows and 20 columns ($A \in \mathbb{R}^{24\times20}$), with 12 rows for upper bounds on $x_t^R, x_t^S, s_t^R, and\ s_t^S$, and 12 rows for positivity constraints on $x_t^R, x_t^S, s_t^R, and\ s_t^S$. The vector b corresponds to the matrix vector product of Ax ($b \in \mathbb{R}^{24}$).

The matrix of equality constraints B has 14 rows and 20 columns ($B \in \mathbb{R}^{14\times20}$) for the combined case. There are 6 rows for demand schedule constraints for the retailer and supplier. There are an additional 6 rows to set $y_t^R, y_t^S = 1$ for $t = \{1,2,3\}$ as part of the order acceptance simplification. The last two rows set the initial storage to zero since analysis assumes zero initial storage in the first month. The combined case matrices and vectors are now solved by linprog. The vector d is the set of corresponding results to the product of Bx ($d \in \mathbb{R}^{14}$).

This gives a new result of \$1546 for the combined cost to the retailer and the supplier.

Sensitivity Analysis

A sensitivity analysis was performed on all parameters using the combined case to find final cost.

First, the additional upper bounds on variables were varied. Table 8 shows the sensitivity analysis for changing the upper bounds on x_t^R and s_t^R. Further constraining maximum orders on the retailer leads to an infeasible problem but increasing the upper bound does not change the value of the optimal solution. This follows logically, as orders are already minimized to avoid excess product and unnecessary storage costs in a month (demand and consumption are still the same). Decreasing the storage upper bound also leads to an infeasible problem and increasing the upper bound leads to a decreased cost. This is because allowing the retailer to store goods is cheaper than the supplier storing goods due to the supplier's higher storage cost. The ordering schedule of the retailer shifts from $x^R = \{40,50,60\}$ to $x^R = \{40,55,55\}$. The retailer can choose to store an additional 5 units in the second month through which the supplier can save more on cost and repay the retailer via side payment.

Table 9 shows the sensitivity analysis for changing the upper bounds on x_t^S and s_t^S. Decreasing the order limit of the supplier leads to an infeasible problem or to a higher cost due to additional storage. Increasing the order limit leads to a lower cost. This is because the supplier no longer needs to plan storage as the ideal replenishment plan of the retailer can be met exactly each month, without needing to store any products.

x_t^R: Upper Bound	Cost	s_t^R: Upper Bound	Cost
50	Infeasible	20	Infeasible
55	Infeasible	25	Infeasible
60	**1546**	**30**	**1546**
65	1546	35	1511
70	1546	40	1511
75	1546	45	1511

Table 8: Sensitivity Analysis on Upper Bounds for Retailer on Combined Case

x_t^S: Upper Bound	Cost	s_t^S: Upper Bound	Cost
45	Infeasible	0	Infeasible
50	1711	5	1546
55	**1546**	**10**	**1546**
60	1451	15	1546
65	1451	20	1546
70	1451	25	1546

Table 9: Sensitivity Analysis on Upper Bounds for Supplier on Combined Case

A sensitivity analysis of the combined case was performed with respect to the demand from the market to the retailer. The demand for each month and the total cost for the combined

case is given in Table 10. It is important to note that this analysis only looks at costs to the companies, and that the net profit for the companies will depend on marginal revenue as well. A higher production leads to a higher cost, but also increased revenue for the company. A higher demand would otherwise not be pursued. The values for orders x_1^S and x_1^R both increase, which increases the total cost. A sensitivity analysis of the price with d_1 is linear as given in Figure 1.

d_1	Cost	d_2	Cost	d_3	Cost
5	$1,506	45	$1,486	85	$1,431
6	$1,514	46	$1,498	86	$1,454
7	$1,522	47	$1,510	87	$1,477
8	$1,530	48	$1,522	88	$1,500
9	$1,538	49	$1,534	89	$1,523
10	$1,546	50	$1,546	90	$1,546
11	$1,554	51	$1,565	91	Infeasible
12	$1,562	52	$1,584	92	Infeasible
13	$1,570	53	$1,603	93	Infeasible
14	$1,578	54	$1,622	94	Infeasible
15	$1,586	55	$1,641	95	Infeasible

Table 10: Sensitivity Analysis of Total Cost Based on Market Demand

Analysis of d_2 shows a slightly different pattern, as given in Figure 2. Initially, there is a linear increase in price due to more orders by the retailer and supplier up to 50 units. After this point, the price continues to grow linearly but with a higher slope. This is because the additional units now require extra storage as well, which increases total unit cost of units as well as total storage cost. Analysis of d_3, given in Figure 3, shows a linear plot with a drop-off due to infeasible points. The plot shows a linear relationship with a steeper slope than Figure 1, as nearly all changes in cost involve changes in both storage costs and unit costs. The plot reaches an infeasible point when demand exceeds 90 in the last month as there is no longer enough storage to meet demand ($s_2^R = 30\ and\ x_3^R = 60$ are maximum values).

A final analysis was performed to the combined case cost with respect to the shipping cost, unit cost, and storage cost to the retailer and supplier. Figure 4 shows a plot of the sensitivity analysis for costs ranging from $1 to $15 for each cost. Shipping costs are denoted f_r and f_s for the retailer and supplier, respectively. Unit costs are denoted p_r and p_s for the retailer and supplier, respectively. Storage costs are denoted h_r and h_s for the retailer and supplier, respectively.

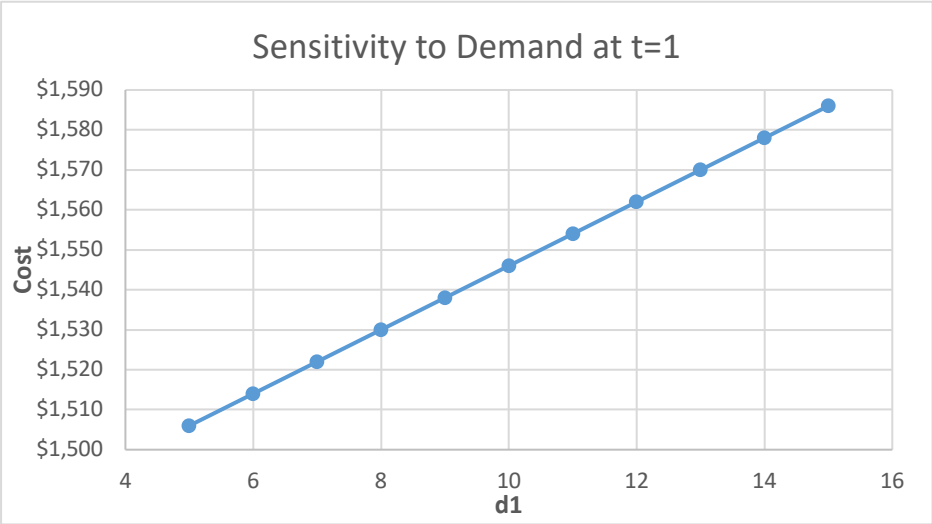

Figure 1: Sensitivity Analysis of Combined Cost with d_1

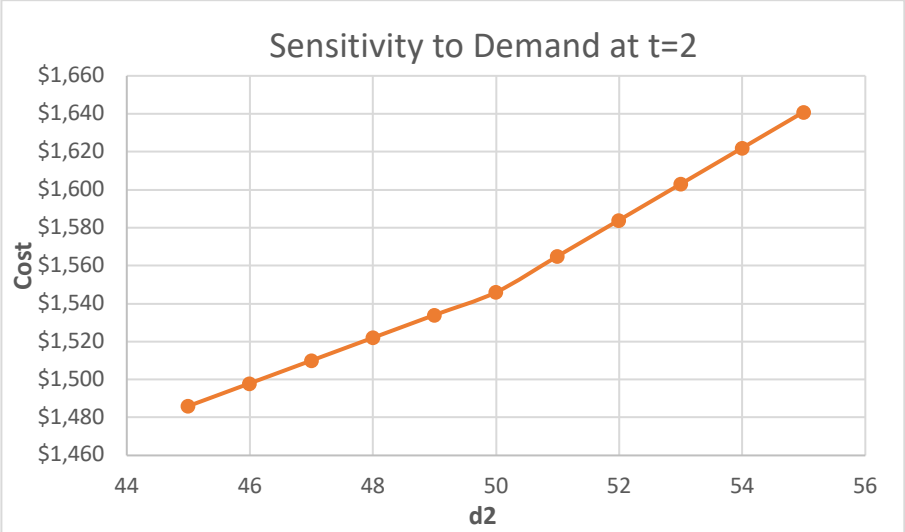

Figure 2: Sensitivity Analysis of Combined Cost with d_2

There are three plots that appear nearly flat compared to others, showing little change in overall cost with ranging prices. There are the two shipping costs as well as the storage cost to the supplier. Shipping costs are unavoidable to the retailer and supplier, but these are only charged 3 times. This expectedly does not increase cost significantly. The storage cost does not change price significantly as the supplier does not require too much storage based on the demand schedule and so the linear relation appears to be flat.

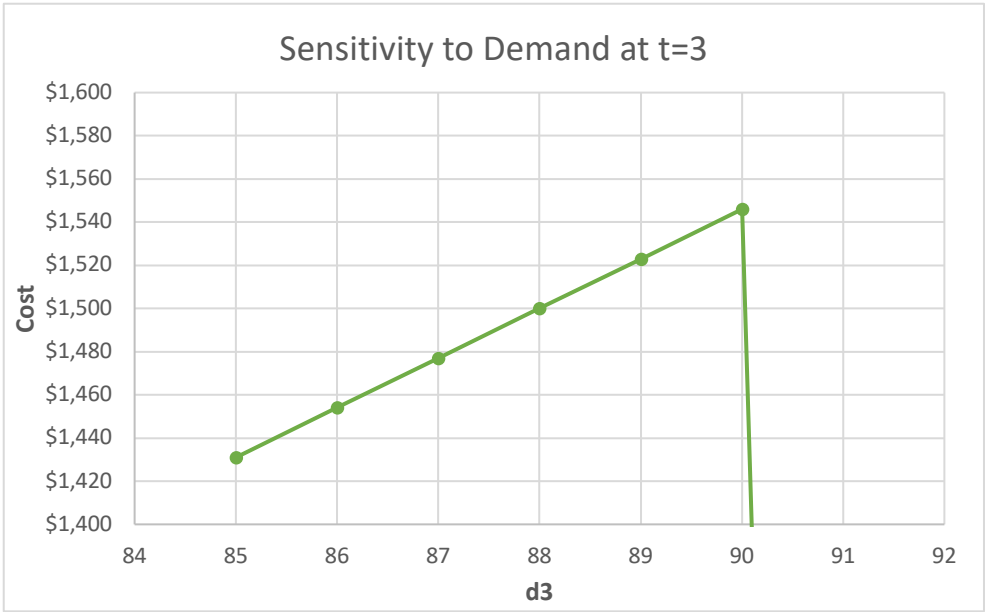

Figure 3: Sensitivity Analysis of Combined Cost with d_3

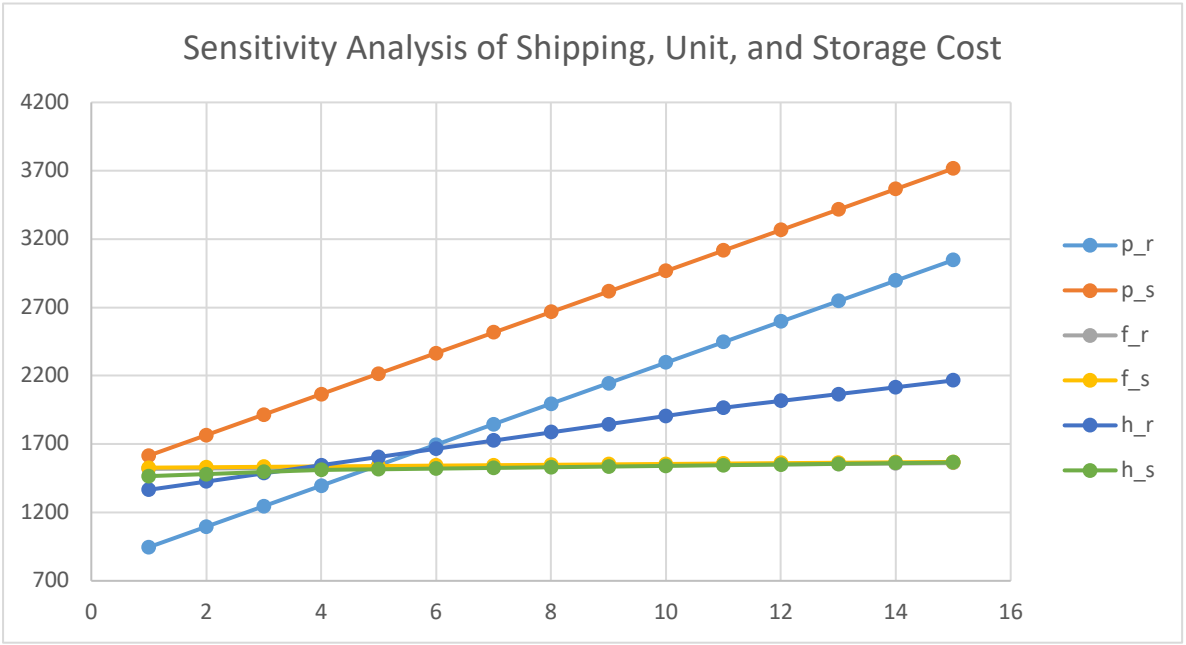

Figure 4: Sensitivity Analysis for Price Changes during Production

Increases in unit prices and storage costs for the retailer change the overall price linearly. Increasing unit price increases the overall price quickly for both cases as the total demand stays the same, forcing production to be the same while increasing total cost. Storage costs to the supplier also increase linearly with storage cost. This is because the retailer must store more units in order to meet market demand.

Conclusion

The original posed problem is a comparison of symmetric and asymmetric information sharing between two companies on the final cost of production. The results of the combined case show that the two companies indeed benefit from cooperation and symmetric information sharing with an optimal value of $1546 as compared to the asymmetric solution, which gives an optimal value of $1616. The retailer's decision to share information drove 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.

The decrease in cost over 3 months is $1616-$1546 = $70. This decrease in cost was calculated for a set of small scale orders and a small time period (T=3). For a larger company, with a production scaled up by a factor 10,000, demands will increase from 10, 50, and 90 over three months to 10000, 50000, and 90000 units of product, which will in turn (assuming other limits also scale up) have a decrease in cost of $700,000 from information sharing, which is significant for a single product. Difference in 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.

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. These additions can potentially make the objective function into an NLP. Changing prices for shipping and per unit of product could also be modeled over time. With certain prediction functions for expected prices the cast can be optimized for both companies.

One method to make transactions more realistic would be to lift the restriction that orders are made every month. This would turn the current LP into a BILP as $y_t \in \{0,1\}$. Another potential issue is the ordering and selling of two products, where the supplier has a finite number of workers to produce either of the two products. In this situation, the production of one product is tied to another and adds another constraint to the problem.

References

[1]Treasury Today. Introduction to Financial Supply Chain Management. http://treasurytoday.com/2007/01/introduction-to-financial-supply-chain-management (accessed November 11, 2018).
[2]Investopedia. https://www.investopedia.com/terms/e/economicorderquantity.asp (accessed November 11, 2018).
[3] 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).
[4] 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).
[5] Allied Shirts. https://www.alliedshirts.com/designer?cppid=133&quantity=1&colorId=260 (accessed November 11, 2018).
[6] scipy.optimize.linprog. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html (accessed Nov 13, 2018).

Appendices

The following set below contains all solutions to the optimization problem in the computational setup section. The set is an intersection between a collection of half spaces (polytope) and an n-m dimensional hyperplane.

$$X = \{x \epsilon \mathbb{R}^n : Ax \leq b\, , Bx = d\}$$

$$Ax \leq b$$
$$A(\alpha x_1 + (1-\alpha)x_2) \leq b \quad \alpha \epsilon\, [0,1]$$
$$\alpha A x_1 + A x_2 - \alpha A x_2 \leq b$$
$$\alpha A x_1 + A x_2 - \alpha A x_2 \leq \alpha b + b - \alpha b \leq b$$
$$b \leq b$$

$$Bx = d$$
$$B(\alpha x_1 + (1-\alpha)x_2) = d \quad \alpha \epsilon\, [0,1]$$
$$\alpha B x_1 + B x_2 - \alpha B x_2 = d$$
$$\alpha d + d - \alpha d = d$$
$$d = d$$

x_1, x_2 are two points that live in the convex set

Because both sets are convex the intersection of two convex sets is also convex.

Ryan Jaipersaud, Armaan Thapar

November 13, 2018

Convex Optimization

Retailer Solution

In [1]:
```
import numpy as np
from scipy.optimize import linprog


# Inequalities Matrix
A = np.array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
              [-1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, -1, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, -1, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, -1, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, -1, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, -1],
              [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])
# Results vector for inequalities
b = np.array([60, 60, 60, 0, 0, 0, 0, 0, 0, 30, 30, 30])
```

In [2]:
```
# Matrix of equalities
B = np.array([[1, 0, 0, 0, 0, 0, 1, -1, 0, 0],
              [0, 1, 0, 0, 0, 0, 0, 1, -1, 0],
              [0, 0, 1, 0, 0, 0, 0, 0, 1, -1],
              [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]])

# Results vector for equalities
d = np.array([10, 50, 90, 1, 1, 1, 0])
```

In [3]:
```
# Cost vector
c= [5, 5, 5, 10, 10, 10, 4, 4, 4, 4]

# Calculate Solution
sol = linprog(c, A, b, B, d)

# Print Solution
print(sol)
```

```
     fun: 980.0
 message: 'Optimization terminated successfully.'
     nit: 9
   slack: array([ 30.,   0.,   0.,  30.,  60.,  60.,  20.,  30.,   0.,  1
0.,   0.,
        30.])
  status: 0
 success: True
       x: array([ 30.,  60.,  60.,   1.,   1.,   1.,   0.,  20.,  30.,
0.])
```

Ryan Jaipersaud, Armaan Thapar

November 13, 2018

Convex Optimization

Supplier Solution

In [6]:
```
import numpy as np
from scipy.optimize import linprog
```

In [7]:
```
# Inequalities Matrix
A = np.array([
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
    ])

# Results vector for inequalities
b = np.array([55, 55, 55, 0, 0, 0, 0, 0, 0, 10, 10, 10])
```

In [8]:
```
# Matrix of equalities
B = np.array([
    [-1, 0, 0, 1, 0, 0, 0, 0, 0, 1, -1, 0, 0],
    [0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1, -1, 0],
    [0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1, -1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ])

# Results vector for equalities
d = np.array([0, 0, 0, 1, 1, 1, 0, 30, 60, 60])
```

In [9]:
```
# Cost vector
c = np.array([0, 0, 0, 3, 3, 3, 7, 7, 7, 0, 11, 11, 11])

# Calculate Solution
sol = linprog(c, A, b, B, d)

# Print Solution
print(sol)
```

```
     fun: 636.0
 message: 'Optimization terminated successfully.'
     nit: 12
   slack: array([ 15.,   0.,   0.,  40.,  55.,  55.,  10.,   5.,   0.,
0.,   5.,
        10.])
  status: 0
 success: True
       x: array([ 30.,  60.,  60.,  40.,  55.,  55.,   1.,   1.,   1.,
0.,  10.,
         5.,   0.])
```

Ryan Jaipersaud, Armaan Thapar

November 13, 2018

Convex Optimization

Combined Case Solution

In [1]:
```
import numpy as np
from scipy.optimize import linprog
```

In [2]:
```
# Inequalities Matrix
A = np.array([
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
    ])

# Results vector for inequalities
b = np.array([60, 60, 60, 55, 55, 55, 0, 0, 0, 0, 0, 0,
              0, 0, 0, 0, 0, 0, 30, 30, 30, 10, 10, 10])
```

In [3]:
```
# Matrix of equalities
B = np.array([
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0],
    [-1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0],
    [0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0],
    [0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
])

# Results vector for equalities
d = np.array([10, 50, 90, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0])
```

In [4]:
```
# Cost vector
c = np.array([5, 5, 5, 3, 3, 3, 10, 10, 10, 7, 7, 7, 0, 4, 4, 4, 0, 11, 11,

# Calculate Solution
sol = linprog(c, A, b, B, d)

# Print Solution
print(sol)
```

```
     fun: 1546.0
 message: 'Optimization terminated successfully.'
     nit: 17
   slack: array([ 20.,  10.,   0.,  15.,   0.,   0.,  40.,  50.,  60.,  4
0.,  55.,
        55.,  30.,  30.,   0.,   0.,   5.,   0.,   0.,   0.,  30.,  10.,
         5.,  10.])
  status: 0
 success: True
       x: array([ 40.,  50.,  60.,  40.,  55.,  55.,   1.,   1.,   1.,
1.,   1.,
         1.,   0.,  30.,  30.,   0.,   0.,   0.,   5.,   0.])
```