Gerald Jones

Individual Project 4:

ISE522 Spg 22

Notebook Links:

1. Data Display section: Display of Data for warehouses and customers
2. Model Formulation: Mathematical formulation of problem
3. Method Definitions: Python code for various tasks
4. Gurobi Implementation: Definition and omptimization with python and Gurobi
5. Solution Discussion: A discussion and explanation of the solution

Problem Description:

Problem Description:

> The attached spreadsheet contains demand data for **ten weeks**. You are to solve the following replenishment problem. **Decide how much to order each week to minimize the total cost**, which includes **fixed charges for ordering and holding cost**.
>
> **Inventory held at** the end of the week at the **supplier and** at our **warehouse incurs a holding cost per unit**. There is a **minimum amount of inventory that must be on hand each week at the warehouse**. If an order is placed in a period, the **order amount must be at least the minimum order quantity (MOQ)**. **When purchasing**, there is a **quantity discount**, as shown in the spreadsheet shown in the [Data Display section](#Data-Display).
>
> - **Inventory can only be held at the supplier for a certain number of weeks.**
>   - **when an order is made it can be held at the supplier instead of immediatly shipped**
>   - **need a decision var to indicate when to ship and when to order**
>   - **After that, the inventory is automatically shipped to the warehouse.**
>   - **If a shipment is made, it must be at least the minimum shipment quantity (MSQ).**
>
> **The attached spreadsheet includes all of the data that you need as well as an example solution. (This example solution is not necessarily optimal. You need to find the optimal solution.**

Notes and Observations

- **assuming infinite capacity for warehouse?**
- **do orders from supplier stack, i.e. if an order is made in week 1 and another order is made in week 2 when order one has been in held for the maximum weeks does all of the held amount get shipped or only the amount that has been held for the given limit?**
- **can make an order, and leave it at supplier for storage for some set amount of weeks, then must push to Warehouse**
- **warehouse & supplier storage costs equivalent?**
  - **No!:**
    - **supplier storage is cheaper by half**
- **Decisions that need to be made each week**
  - **how much to order**
  - **ship the order or store it at supplier**
  - **how much to pull from supplier**
  - **if we have to pull from supplier what was already stored there**
- **Costs Depend on:**
  - **how much we hold at warehouse**
  - **how much we hold at supplier**
- **Goal: Minimize cost after the ten weeks forcast**

Assumptions:

- **when an order amount is left at the supplier at time t, at time t+3 the amount left overe from that order needs to be shipped to warehouse**

Module imports and data loading

In [1]:
```python
from _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
from _NOTE_BOOK_UTILS import *
import numpy as np


from  _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
from _NOTE_BOOK_UTILS import *
# name of notebook used to generate a pdf
notebook_name = "_IP4_Gerald_Jones.ipynb"

# create short hand versions of the column names we will need
Dem = "Demand"
MIR = "Minimum inventory requirement"

filename = "single item data - MOQ and qty discount and VMI v2.xlsx"
data_cols = [Dem, MIR]

# full data
data_df_full = pd.read_excel(filename)

# only grab the data we will need for the model
data_df = data_df_full.iloc[list(range(9, 19)), :]

# grab the appropriate column names and set them as the data frames columns names
data_df.columns = data_df_full.iloc[7, :].tolist()
# set the indices to match we expect
data_df.index = list(range(data_df.shape[0]))
data_df = data_df.filter(items=data_cols)

weekly_demands = data_df.loc[:, Dem].values.tolist()
weekly_MIR = data_df.loc[:, MIR].values.tolist()

# set minimum order quantity
MOQ = data_df_full.loc[0, 5]
M = data_df["Demand"].sum()
MaxWeeksHold = data_df_full.loc[1, 5]

# add Big M value to ranges
# makes it easier to set up constraints in the way I have done it
discount_ranges = data_df_full.iloc[4, 1:4].values.tolist() + [M]
discount_prices = data_df_full.iloc[5, 1:4].values

# set up the initial amounts of units on hand for each location
Hw0 = data_df_full.iloc[8, 5]
Hs0 = 0

# set up the holding costs for the two locations
# since they are constant just set as variables
HoldingCostW = 2.0
HoldingCostS = 1.0
R = 3
```

Data Display

In [2]:
```python
# display data for problem
print("\t\tFull Data File")
uniPrint(data_df_full)
print("\t\tWeekly Warehouse, Supplier, Demand Data")
uniPrint(data_df)
print("weekly demands:")
print(weekly_demands)
print("weekly minimum order requirements")
print(weekly_MIR)
```

Full Data File

	min order qty	5	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unname 1
0	min shipment qty	5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	max number of weeks that inventory can be held...	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	Quantity discount schedule	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	NaN	minimum order amount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	5	20	30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	price per unit	3	1	0.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7	Week	Demand	Order amount	Shipment amount from supplier to warehouse	inventory at supplier at end of week	Inventory at warehouse end of week	Minimum inventory requirement	Holding cost per unit	Holding cost per unit at supplier	Cost of goods purchased	Holding cost - warehouse	Holdin cost suppli
8	0	NaN	NaN	NaN	NaN	20	NaN	NaN	NaN	NaN	NaN	Na
9	1	30	15	NaN	NaN	5	4.5	2	1	45	10	
10	2	0	0	NaN	NaN	5	0	2	1	0	10	
11	3	0	10	NaN	NaN	15	0	2	1	30	30	
12	4	0	0	NaN	NaN	15	0	2	1	0	30	
13	5	6	8	NaN	NaN	17	0.9	2	1	24	34	
14	6	4	0	NaN	NaN	13	0.6	2	1	0	26	
15	7	4	0	NaN	NaN	9	0.6	2	1	0	18	
16	8	4	0	NaN	NaN	5	0.6	2	1	0	10	
17	9	4	0	NaN	NaN	1	0.6	2	1	0	2	
18	10	1	0	NaN	NaN	0	0.15	2	1	0	0	
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
20	TOTAL COST	269	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

Weekly Warehouse, Supplier, Demand Data

	Demand	Minimum inventory requirement
0	30	4.5
1	0	0
2	0	0
3	0	0
4	6	0.9
5	4	0.6
6	4	0.6
7	4	0.6
8	4	0.6
9	1	0.15

```
weekly demands:
[30, 0, 0, 0, 6, 4, 4, 4, 4, 1]
weekly minimum order requirements
[4.5, 0, 0, 0, 0.8999999999999999, 0.6, 0.6, 0.6, 0.6, 0.15]
```

Model Formulation

- Paremeters and Sets
- Variables
- Equations and Constraints
- Objective

Parameters and Sets:

$\mathbf{T}$ represents time unit in weeks $t \in T$

w represents index for warehouse

s represents index for supplier

$\mathbf{Q}$ minimum order/shipping quantity

$\mathbf{R}$ set of order ranges for different discount prices $= \{[5,10),[10,15),[15,\mathrm{M}]\}$, $r \in R$

Δ set of discount prices $= \{2,1,.5\}$, $\delta_r \in \Delta$ and δ_r is in discount for order range r

μ_t minimum inventory requirement for week t

$\mathbf{D}_t$ units of demand for week t

ρ_s holding cost for supplier storage

ρ_w holding cost for warehouse storage

M total demand expected over T weeks, large number

τ maximum amount of weeks supplier can hold an order befor shipping to warehouse

Variables:

$X_{t,r}$ amount to order for week t at order amount in range r

Y_t amount to ship from supplier to warehouse in week t

t_o time of order for order $\mathbf{o}$

$O_{t,r}$ ***binary, 1 if ordering in week t for amount in range r, 0 otherwise***

l_t ***binary, 1 if shipping from supplier in week t, 0 otherwise***

$H_{t,w}$ stock on hand at end of week t for warehouse

$H_{t,s}$ stock on hand at end of week w for supplier

P_t binary, 1 if order at placed at time t is still at supplier at time t + 3

S_w minimum stock on hand for week w

U_w per unit cost of stock held at end of week w

O_w binary variable for X_w, representing the decision to order or not

$\mathbf{C_{t,w}}$ warehouse holding costs at week t

$\mathbf{C_{t,\,s}}$ supplier storage costs at week t

C_t total cost in week t

$\mathbf{C}_\mathrm{T}$ total cost after T weeks

Equations and Constraints:

Order Range Variables Constraint

$$R[r]\cdot O_{t,r} \leq X_{t,r} < R[r+1]\cdot O_{t,r}, \forall t, r$$

$$X_{t,r} \leq M\cdot O_{t,r}, \forall t, r$$

$$5\cdot O_{t,0} \leq X_{t,0} < 10\cdot O_{t,0}$$

$$10\cdot O_{t,1} \leq X_{t,1} < 15\cdot O_{t,1}$$

$$15\cdot O_{t,2} \leq X_{t,2} \leq M\cdot O_{t,2}$$

Binary Ordering Decision Variable Expression/Constraints

$$O_{t,r} \in \{0,1\},\ \forall t, r$$

Single Discount Price Constraints

$$\sum_{r=1}^{|R|} O_{t,r} = 1,\ \forall t$$

$$O_{t,r_a} = 1 \implies O_{t,r_b} == 0, \text{such that } r_a, r_b \in R, r_a! = r_b\ \forall t$$

Binary Shipping Decision Variable Expression/Constraints

$$l_t \in \{0,1\},\ \forall t$$

Supply on hand at supplier expressions/constraints

$$H_{t,s} = H_{t-1,s} - Y_t + \sum_0^t X_{t,r}\cdot O_{t,r} - H_{t-3,s}\cdot P_t, \forall t >= 3$$

$$H_{t,s} = H_{t-1,s} - Y_t + \sum_0^t X_{t,r}\cdot O_{t,r}, \forall\ 3 > t > 0$$

$$H_{t,s} = -Y_t + \sum_0^t X_{t,r}\cdot O_{t,r}, t = 0$$

$$H_{t,s} \geq 0$$

$$Q\cdot l_t \leq Y_t \leq m\cdot l_t$$

$$Y_t \geq D_t - H_{t-1,w}$$

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

Supply on hand at warehouse expressions/constraints

$$H_{t,w} = H_{t-1,w} - D_t + Y_t$$

$$H_{t,w} \geq \mu_t$$

Weeks at supplier constraint

if the amount ordered and stored at time t is still in the supplier at time t+3 then we need to add it to what is being shipped

$$P_t == 1 \implies H_{t,s} - H_{t-3,s} >= 0, \forall t \geq 3$$

Cost expressions

$$\mathbf{C}_{t,w} = H_{t,w}\cdot \rho_w$$

$$\mathbf{C}_{t,s} = H_{t,s}\cdot \rho_s$$

$$\mathbf{C}_T = \sum_t^T (C_{t,w} + C_{t,s} + (\sum_r^R X_{t,r}\cdot O_{t,r}\cdot \delta_r))$$

Objective:

$$\min(C_T)$$

Method Definitions

In [3]:
```python
# generate objective
def generate_obj(unit_costS, unit_costW, fr, Ows, Htw, Hts, H0, S0, T):
    expression = 0
    print("cost ware: {}, sup: {}".format(unit_costW, unit_costS))
    for i in range(T):
        order_cost = fr[0]*Ows[i,0] + fr[1]*Ows[i,1] + fr[2]*Ows[i,2]
        supplier_cost = Hts[i]* unit_costS
        warehousing_cost = Htw[i]*unit_costW

        # sum the cost for ordering and holding for each week
        expression += warehousing_cost + order_cost + supplier_cost
    return expression


# generate constraints
```

Gurobi Implementation and Solution

In [4]:
```python
try:
    # instantiate model object
    m = gp.Model("G_MOD")



    ######################################################################################
    ################################# Parameters set up ##################################
    ######################################################################################
    T = len(data_df)
    tau = 4
    params = "\t\tMOQ:{}\n" +\
             "\t\ttau: {}\n" +\
             "\t\t'Big' M: {}\n" +\
             "\t\tT: {}\n" +\
             "\t\tMax Weeks Storage:{}\n" + "\t\tDiscount Order Ranges: {}\n" + "\t\tDiscount Prices: {}\n" +\
             "\t\tInitial amount in warehouse: {}\n" + "\t\tInitial amount as supplier: {}\n" +\
             "\t\tWeekly Demands: {}\n" + "\t\tWeekly Minimum inventory Requirments: {}\n"
    params = params.format(MOQ, tau, M, T, MaxWeeksHold,
                           discount_ranges, discount_prices,
                           Hw0, Hs0,
                           weekly_demands,
                           weekly_MIR)
    print("###############################################################################")
    print(params)
    print("###############################################################################\n")
    ######################################################################################
    ################################# Variables set up ###################################
    ######################################################################################
    # how much to order
    Xtr = m.addVars(T, 3, vtype=GRB.CONTINUOUS, name="OrderAmount_tr", lb=0)

    # controls if an order was mad at week t or not
    Otr = m.addVars(T, 3, vtype=GRB.BINARY, name="OrderDecision_tr")

    # amount on hand at supplier at time t
    Hts = m.addVars(T, vtype=GRB.CONTINUOUS, name="OnHandSupplier_ts", lb=0)

    # amount on hand at warehouse at time t
    Htw = m.addVars(T, vtype=GRB.CONTINUOUS, name="OHandWarehouse_tw")

    # how much to ship
    Yt = m.addVars(T, vtype=GRB.CONTINUOUS, name="ShipAmount_t", lb=0)


    # TODO:  too long holding so ship variable
    Pt = m.addVars(T, vtype=GRB.BINARY, name="HoldingLimitDecision_t", lb=0)

    # TODO:  shipping decision
    lt = m.addVars(T, vtype=GRB.BINARY, name="ShipmentDecision_t", lb=0)

    #                                        Order Amount Constraints
    # constrain each range of order amount to the appropriate ranges
    m.addConstrs( Xtr[t, r] <= (discount_ranges[r+1]-.99)*Otr[t, r] for t in range(T)
                                        for r in range(R))
    m.addConstrs( Xtr[t, r] >= discount_ranges[r]*Otr[t, r]  for t in range(T)
                                        for r in range(R))

    # constrain binary selector variable so that
    # only one is "high" at a time
    for t in range(T):
        expression = 0
        # for each set of ranges
        for r in range(R):
            expression += Otr[t, r]
        m.addConstr(expression <= 1)




    # constrain the shipment mount to shipment decision variable
#     m.addConstrs( (St[t] == 1) >> (Yt[t] >= 0.000009) for t in range(T))   # if shipment occurs set to one
    m.addConstrs( Yt[t] >= MOQ*lt[t] for t in range(T))                     # amount constrained by decision uppe
    m.addConstrs( Yt[t] <= M*lt[t] for t in range(T))                       # amount constrained by decision lowe

    # in first week need to ship at least the difference between what we have and what we need
    m.addConstr(Yt[0] >= weekly_demands[0] - Hw0)

    # for the weeks befor the limit need to send at difference between what is demanded
    # and what is available at the warehouse from last week
    m.addConstrs(Yt[t] >= weekly_demands[t] - Htw[t-1] for t in range(1, 3))

    # if is beyond week limit range start adding the amount that has to be shipped to the amount we need to sh
#     m.addConstrs(Yt[t] >= weekly_demands[t] - Htw[t-1] - Hts[t-3]*Pt[t] for t in range(3, T))
    m.addConstrs(Yt[t] >= weekly_demands[t] - Htw[t-1] for t in range(3, T))

#     for t in range(T):
#         exprs = 0
#         for r in range(R):
#             exprs += Xtr[t, r] * discount_prices[r] * Otr[t, r]
#         if t == 0:
#             m.addConstr(Yt[t] >= weekly_demands[t] - Hw0 )
#         elif t < 3:
# #         else:


    # set the amount to ship to be based on if there was an order in there
    # longer than the limit and the amount on hand
    for t in range(T):
#         print("\n")
#         print("tau-1: {}".format(tau-1))
#         print(t)
#         print(t-(tau-1))
#         print("\n")
        expr = 0
        for r in range(R):
            expr += Xtr[t, r] * Otr[t,r]*discount_prices[r]
        if t ==0:
            # at most can be what was ordered in the first week since none at supplier
            m.addConstr(Yt[t] <= expr)
        elif t < (tau-1):
            # at most can be what is left from last week and what was ordered
            m.addConstr(Yt[t] <= Hts[t-1] + expr)
        else:
#             print(t-tau)
            # beyond the limit range at most send what is available from last time
            # what needs to be shipped since it was there to long and what was ordered
            m.addConstr(Yt[t] <= Hts[t-1] + Hts[t-(tau-1)]*Pt[t] + expr)
#         m.addConstr(Yt[t] >= Hts[t-(tau-1)]*Pt[t])



    # set Pt to be one iff we have enough of the original amount left over to send
    for t in range(T):
        if t <= (tau-1):
#             pass
            # no forcing of shipment until fourth week of analysis
            m.addConstr(Pt[t] == 0)
        else:
            print("t: {}, t-(tau-1): {}".format(t, t-(tau-1)))
            # if the amount on hand at time t minus what we had on hand at time t-tau then we still have what
            # ordered at time t-tau left over so ship it at time t
            m.addConstr( (Pt[t] == 1) >> (Hts[t-1] - Hts[t-(tau-1)] + expression - Yt[t] >= 0.00009) )



    # set up the amount at the supplier constraints
    for t in range(T):
        expression = 0
        for r in range(R):
            expression += Xtr[t, r]*Otr[t,r]

        if t == 0:
            m.addConstr(Hts[t] == expression - Yt[t])
        elif t >= (tau-1):
            # if past the fourth week need to  consider what has been lying in supplier storage
            m.addConstr(Hts[t] == Hts[t-1] + expression - Yt[t] - Hts[t-(tau-1)]*Pt[t])
        else:
            m.addConstr(Hts[t] == Hts[t-1] + expression - Yt[t])


    # constrain amount to ship from supplier to decision variable adn the amount on hand at the time
#     m.addConstrs(Yt[t] <= Hts[t] for t in range(T))



    m.addConstrs(Htw[t] >= weekly_MIR[t] for t in range(T))


    # Number of weeks in storage counter
#     It = m.addVars(T, T, vtype=GRB.CONTINUOUS, name="IterattiveCountert", lb=0)
#     for t in range(T):
#         expr = 0
#         for t2 in range(0, t):
#             expr += It[t2] * Pt[t]
#         m.addConstr(It[t] == expr)
#         m.addConstr((Pt[t] == 1) >> (It[t] >= 4))

    # add expression calculating on hand at warehouse at end of week
    # and add constraint to make sure the min inventory requirement is met
    expression = 0
    for t in range(T):

        if t == 0:
            m.addConstr(Htw[t] == Hw0 + Yt[t] - weekly_demands[t])
        elif t >= (tau-1):
            m.addConstr(Htw[t] == Htw[t-1] + Yt[t] - weekly_demands[t] + Hts[t-(tau-1)]*Pt[t])
        else:
            m.addConstr(Htw[t] == Htw[t-1] + Yt[t] - weekly_demands[t])
        m.addConstr(Htw[t] >= weekly_MIR[t])
    ######################################################################################
    ################################# Objective set up ###################################
    ######################################################################################
    # generate_obj(unit_costS, unit_costW, fr, Ows, Htw, Hts, H0, S0)
    obj_express = generate_obj(HoldingCostS,          # cost of supplier storage
                               HoldingCostW,          # cost of warehousing storage
                               discount_prices,       # fixed rate discounted costs
                               Otr,                   # decision to order in week w
                               Htw,                   # amount held in warehouse by week
                               Hts,                   # amount held by supplier by week
                               H0=Hw0,                # initial amount at warehouse
                               S0=0,                  # initial amount at supplier
                               T=T)                   # the number of time steps

    m.setObjective(obj_express, GRB.MINIMIZE)
    ######################################################################################
    ################################# Constraint set up ##################################
    ######################################################################################


    ######################################################################################
    ################################# SOLVE:OPTIMIZE #####################################
    ######################################################################################


    m.optimize()

    ######################################################################################
    ################################# Display Results ####################################
    ######################################################################################
    displayDecisionVars(m, end_sentinel="9")

    print("\n------------Does it make sense?---------------------")
    print('Obj: {:.2f}'.format(m.ObjVal))


# catch some math errors
except gp.GurobiError as e:
    print('Gurobi-Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

```
Restricted license - for non-production use only - expires 2023-10-25
##########################################################################
                MOQ:5
                tau: 4
                'Big' M: 53
                T: 10
                Max Weeks Storage:4
                Discount Order Ranges: [5, 20, 30, 53]
                Discount Prices: [3 1 0.5]
                Initial amount in warehouse: 20
                Initial amount as supplier: 0
                Weekly Demands: [30, 0, 0, 0, 6, 4, 4, 4, 4, 1]
                Weekly Minimum inventory Requirments: [4.5, 0, 0, 0, 0.8999999999999999, 0.6, 0.6, 0.6, 0.6,
0.15]

##########################################################################

t: 4, t-(tau-1): 1
t: 5, t-(tau-1): 2
t: 6, t-(tau-1): 3
t: 7, t-(tau-1): 4
t: 8, t-(tau-1): 5
t: 9, t-(tau-1): 6
cost ware: 2.0, sup: 1.0
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 127 rows, 110 columns and 241 nonzeros
Model fingerprint: 0x1a9e2c6c
Model has 27 quadratic constraints
Model has 6 general constraints
Variable types: 60 continuous, 50 integer (50 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  QMatrix range    [5e-01, 3e+00]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [5e-01, 3e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e-01, 1e+01]
  QRHS range       [1e+00, 6e+00]
  GenCon rhs range [9e-05, 9e-05]
  GenCon coe range [1e+00, 1e+00]
Presolve removed 26 rows and 8 columns
Presolve time: 0.00s
Presolved: 235 rows, 138 columns, 647 nonzeros
Variable types: 93 continuous, 45 integer (45 binary)
Found heuristic solution: objective 258.8500000

Root relaxation: objective 4.626087e+01, 54 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   46.26087    0   14  258.85000   46.26087  82.1%     -    0s
H    0     0                     238.0000000   46.26087  80.6%     -    0s
     0     0   60.47671    0   23  238.00000   60.47671  74.6%     -    0s
H    0     0                     116.2000000   60.47671  48.0%     -    0s
     0     0   60.47671    0   24  116.20000   60.47671  48.0%     -    0s
     0     0   67.61515    0   21  116.20000   67.61515  41.8%     -    0s
H    0     0                     104.3000000   67.61515  35.2%     -    0s
H    0     0                     101.1500000   67.61515  33.2%     -    0s
     0     0   68.97939    0   23  101.15000   68.97939  31.8%     -    0s
     0     0   69.02666    0   24  101.15000   69.02666  31.8%     -    0s
H    0     0                      89.3500000   69.02666  22.7%     -    0s
H    0     0                      79.3002700   69.02666  13.0%     -    0s
     0     0   70.44636    0   19   79.30027   70.44636  11.2%     -    0s
H    0     0                      73.2004500   70.44636  3.76%     -    0s
     0     0   70.53892    0   18   73.20045   70.53892  3.64%     -    0s
*    0     0               0      73.2000000   73.20000  0.00%     -    0s

Cutting planes:
  Gomory: 2
  Implied bound: 7
  MIR: 30
  Flow cover: 23
  Relax-and-lift: 4

Explored 1 nodes (353 simplex iterations) in 0.07 seconds (0.01 work units)
Thread count was 12 (of 12 available processors)

Solution count 9: 73.2 73.2004 79.3003 ... 258.85

Optimal solution found (tolerance 1.00e-04)
Warning: max constraint violation (9.0000e-05) exceeds tolerance
Warning: max general constraint violation (9.0000e-05) exceeds tolerance
Best objective 7.320000000000e+01, best bound 7.320000000000e+01, gap 0.0000%
OrderAmount_tr[0,0] 14.50000
OrderAmount_tr[0,1] 0.00000
OrderAmount_tr[0,2] 0.00000
OrderAmount_tr[1,0] 0.00000
OrderAmount_tr[1,1] 0.00000
OrderAmount_tr[1,2] 0.00000
OrderAmount_tr[2,0] 0.00000
OrderAmount_tr[2,1] 0.00000
OrderAmount_tr[2,2] 0.00000
OrderAmount_tr[3,0] 0.00000
OrderAmount_tr[3,1] 0.00000
OrderAmount_tr[3,2] 0.00000
OrderAmount_tr[4,0] 6.10000
OrderAmount_tr[4,1] 0.00000
OrderAmount_tr[4,2] 0.00000
OrderAmount_tr[5,0] 0.00000
OrderAmount_tr[5,1] 0.00000
OrderAmount_tr[5,2] 0.00000
OrderAmount_tr[6,0] 8.00000
OrderAmount_tr[6,1] 0.00000
OrderAmount_tr[6,2] 0.00000
OrderAmount_tr[7,0] 0.00000
OrderAmount_tr[7,1] 0.00000
OrderAmount_tr[7,2] 0.00000
OrderAmount_tr[8,0] 5.00000
OrderAmount_tr[8,1] 0.00000
OrderAmount_tr[8,2] 0.00000
OrderAmount_tr[9,0] 0.00000

OrderAmount_tr[9,1] 0.00000

OrderAmount_tr[9,2] 0.00000

OrderDecision_tr[0,0] 1.00000
OrderDecision_tr[0,1] -0.00000
OrderDecision_tr[0,2] 0.00000
OrderDecision_tr[1,0] -0.00000
OrderDecision_tr[1,1] -0.00000
OrderDecision_tr[1,2] -0.00000
OrderDecision_tr[2,0] 0.00000
OrderDecision_tr[2,1] 0.00000
OrderDecision_tr[2,2] -0.00000
OrderDecision_tr[3,0] -0.00000
OrderDecision_tr[3,1] -0.00000
OrderDecision_tr[3,2] -0.00000
OrderDecision_tr[4,0] 1.00000
OrderDecision_tr[4,1] 0.00000
OrderDecision_tr[4,2] -0.00000
OrderDecision_tr[5,0] -0.00000
OrderDecision_tr[5,1] -0.00000
OrderDecision_tr[5,2] -0.00000
OrderDecision_tr[6,0] 1.00000
OrderDecision_tr[6,1] -0.00000
OrderDecision_tr[6,2] -0.00000
OrderDecision_tr[7,0] 0.00000
OrderDecision_tr[7,1] -0.00000
OrderDecision_tr[7,2] 0.00000
OrderDecision_tr[8,0] 1.00000
OrderDecision_tr[8,1] -0.00000
OrderDecision_tr[8,2] -0.00000
OrderDecision_tr[9,0] -0.00000

OrderDecision_tr[9,1] -0.00000

OrderDecision_tr[9,2] -0.00000

OnHandSupplier_ts[0] 0.00000
OnHandSupplier_ts[1] 0.00000
OnHandSupplier_ts[2] 0.00000
OnHandSupplier_ts[3] 0.00000
OnHandSupplier_ts[4] 0.00000
OnHandSupplier_ts[5] 0.00000
OnHandSupplier_ts[6] 0.00000
OnHandSupplier_ts[7] 0.00000
OnHandSupplier_ts[8] 0.00000
OnHandSupplier_ts[9] 0.00000

OHandWarehouse_tw[0] 4.50000
OHandWarehouse_tw[1] 4.50000
OHandWarehouse_tw[2] 4.50000
OHandWarehouse_tw[3] 4.50000
OHandWarehouse_tw[4] 4.60000
OHandWarehouse_tw[5] 0.60000
OHandWarehouse_tw[6] 4.60000
OHandWarehouse_tw[7] 0.60000
OHandWarehouse_tw[8] 1.60000
OHandWarehouse_tw[9] 0.60000

ShipAmount_t[0] 14.50000
ShipAmount_t[1] 0.00000
ShipAmount_t[2] 0.00000
ShipAmount_t[3] 0.00000
ShipAmount_t[4] 6.10000
ShipAmount_t[5] 0.00000
ShipAmount_t[6] 8.00000
ShipAmount_t[7] 0.00000
ShipAmount_t[8] 5.00000
ShipAmount_t[9] -0.00000

HoldingLimitDecision_t[0] 0.00000
HoldingLimitDecision_t[1] 0.00000
HoldingLimitDecision_t[2] 0.00000
HoldingLimitDecision_t[3] 0.00000
HoldingLimitDecision_t[4] -0.00000
HoldingLimitDecision_t[5] 0.00000
HoldingLimitDecision_t[6] -0.00000
HoldingLimitDecision_t[7] -0.00000
HoldingLimitDecision_t[8] 0.00000
HoldingLimitDecision_t[9] 1.00000

ShipmentDecision_t[0] 1.00000
ShipmentDecision_t[1] 0.00000
ShipmentDecision_t[2] 0.00000
ShipmentDecision_t[3] 0.00000
ShipmentDecision_t[4] 1.00000
ShipmentDecision_t[5] 0.00000
ShipmentDecision_t[6] 1.00000
ShipmentDecision_t[7] 0.00000
ShipmentDecision_t[8] 1.00000
ShipmentDecision_t[9] -0.00000

------------------------------------



-------------Does it make sense?----------------------
Obj: 73.20
```

Solution Discussion

> **The solution....**
>
> **I expected the solution to try to have as much in supplier storage as possible, only order as much as needed to cover the expected demands and only ship to the warehouse what they need. The solution does this but does not utilize the supplier storage at all. The solution orders an (week 1) initial 14.5 to go along with its already available 20 units to and ships that so that the warehouse has an initial total value of 34.5 and with the initial demand of 30 that leaves exactly 4.5 which is the minimum stock amount for that week. The next order occurs at the fifth week were an order of 6.1 units. is made and immediately shipped to the warehouse so that the 6 unit demand can be met leaving 4.6 units at the warehouse. The next order is made at week 7 of 8 units to meet the 4 unit demand at week 7 leaving 4.6 units at the warehouse. The next order is for 5 units in week 9 to meet the 4 unit demand in that week leaving 1.6 units at the warehouse in week 9. Finally the week 10 demand of 1 unit is met with the remaining 1.6 units at the warehouse leaving only .6 units in the warehouse. The suggested solution does meet all required demands, the minimum inventory requirements for each week, and the minimum order/shipping requirements. The supposed optimal solution is an overall cost of $73.20K I feel there may be some error since the solution does not try to utilize the lower storage costs at the supplier at all.**

In [6]:
```
#save the notebook as a pdf
to_PDF(notebook_name)
```

```
filename: _IP4_Gerald_Jones.ipynb
```