Aggregate planning (production and inventory models)

Andrés D. González

Assistant Professor
School of Industrial and Systems Engineering, The University of Oklahoma

ISE 4623/5023: Deterministic Systems Models / Systems Optimization

The University of Oklahoma, Norman, OK, USA

The aggregate planning problem

- Aggregate planning:
  - Process by which a company determines levels of capacity, production, subcontracting, inventory, stockouts, and pricing over a specified time horizon
  - Goal is to maximize profit
  - Decisions made at a product family (not SKU) level
  - Time frame of 3 to 18 months
  - How can a firm best use the facilities it has?
- Identify operational parameters over the specified time horizon
  - Production rate
  - Workforce
  - Overtime
  - Machine capacity level
  - Subcontracting
  - Backlog
  - Inventory on hand

The aggregate planning problem

- Given the demand forecast for each period in the planning horizon, determine the production level, inventory level, and the capacity level for each period that maximizes the firm's (supply chain's) profit over the planning horizon
  - Specify the planning horizon (typically 3-18 months)
  - Specify the duration of each period
  - Specify key information required to develop an aggregate plan

Information needed for aggregate plan

- Aggregate demand forecast F_t for each Period t over T periods
- Production costs
  - Labor costs, regular time ($/hr) and overtime ($/hr)
  - Subcontracting costs ($/hr or $/unit)
  - Cost of changing capacity – hiring or layoff ($/worker), adding or reducing machine capacity ($/machine)
- Labor/machine hours required per unit
- Inventory holding cost ($/unit/period)
- Stockout or backlog cost ($/unit/period)
- Constraints – overtime, layoffs, capital available, stockouts, backlogs, from suppliers

Outputs of aggregate plan

- Production quantity from regular time, overtime, and subcontracted time
- Inventory held
- Backlog/stockout quantity
- Workforce hired/laid off
- Machine capacity increase/decrease

A poor aggregate plan can result in lost sales, lost profits, excess inventory, or excess capacity

Aggregate plan/inventory model – sets

Set	Definition / description
T	Set of months (or periods) in the planning horizon
L	Set of commodities / products

Aggregate plan/inventory model - parameters

Param.	Definition / description
d_{lt}	Demand of commodity l in period t
p_{lt}	Material / unit production cost of commodity l
i_{lt}	Inventory holding cost of commodity l
s_l	Marginal cost of stockout/backlog of commodity l
h	Hiring and training costs per worker
f	Layoff cost per worker
k_l	Labor hours required per unit of commodity l

Param.	Definition / description
w	Regular time cost (per hour per worker)
o	Overtime cost (per hour)
c_l	Cost of subcontracting a unit of commodity l
a_l	Initial inventory of commodity l
b	Initial workforce
e_l	Initial backlog of commodity l
n_t	Number of regular working hours in month t
m_t	Maximum number of overtime hours per worker in month t

Aggregate plan/inventory model – decision variables

Likely will always be an int

Could be either an integer or double

Variable	Definition / description
W_t	Workforce size for month t
H_t	Number of employees hired at the beginning of month t
F_t	Number of employees laid off at the beginning of month t
P_{lt}	Production in month t of commodity l
I_{lt}	Inventory at the end of month t of commodity l
S_{lt}	Number of units stocked out at the end of month t of commodity l
C_{lt}	Number of units subcontracted for month t of commodity l
O_t	Number of overtime hours worked in month t

Aggregate plan/inventory model – objective function

- Minimize total cost:

  (Regular-time labor cost) + (Overtime labor cost) + (Cost of hiring) + (Cost of layoffs) + (Cost of holding inventory) + (Cost of stocking out) + (Material cost) + (Cost of subcontracting)

$$\min \quad \sum_{t\in T} n_t w W_t + \sum_{t\in T} o O_t + \sum_{t\in T} h H_t + \sum_{t\in T} f F_t + \sum_{t\in T}\sum_{l\in L} i_{lt} I_{lt} + \sum_{t\in T}\sum_{l\in L} s_l S_{lt} + \sum_{t\in T}\sum_{l\in L} p_{lt} P_{lt} + \sum_{t\in T}\sum_{l\in L} c_l C_{lt}$$

Aggregate plan/inventory model – constraints

Workforce, hiring, and layoff constraints

$$W_1 = b + H_1 - L_1$$

$$W_t = W_{t-1} + H_t - L_t \qquad \forall t \in T \setminus \{1\}$$

Capacity constraints

$$\sum_{l \in L} k_l P_{lt} \leq n_t W_t + O_t \qquad \forall t \in T$$

Inventory balance constraints

$$a_l + P_{l1} + C_{l1} - e_l = D_{l1} + I_{l1} - S_{l1} \qquad \forall l \in L$$

$$I_{l,t-1} + P_{lt} + C_{lt} - S_{l,t-1} = D_{lt} + I_{lt} - S_{lt} \qquad \forall l \in L, \forall t \in T \setminus \{1\}$$

Overtime limit constraints

$$O_t \leq m_t W_t \qquad \forall t \in T$$

Nature of the variables

$$W_t, H_t, F_t, O_t \in \mathbb{Z}^+ \cup \{0\} \qquad \forall t \in T$$

$$P_{lt}, I_{lt}, S_{lt}, C_{lt} \geq 0 \qquad \forall l \in L, \forall t \in T$$

Example – Single Commodity - Red Tomato Tools

Month	Demand Forecast
January	1,600
February	3,000
March	3,200
April	3,800
May	2,200
June	2,200

Item	Cost
Material cost	$10/unit
Inventory holding cost	$2/unit/month
Marginal cost of stockout/backlog	$5/unit/month
Hiring and training costs	$300/worker
Layoff cost	$500/worker
Labor hours required	4/unit
Regular time cost	$4/hour
Overtime cost	$6/hour
Cost of subcontracting	$30/unit
Initial inventory	1000
Initial workforce	80

Building and solving the model with Gurobi/Python

```
from gurobipy import *
model=Model('Inventory')

#Sets and parameters
p=10
i=2
s=5
h=300
f=500
k=4
w=4
o=6
c=30
a=1000
b=80
e=0
n=160
m=10

T,d=multidict({
    (1):1600,
    (2):3000,
    (3):3200,
    (4):3800,
    (5):2200,
    (6):2200
})

#Variables
W=model.addVars(T, obj=n*w,vtype=GRB.INTEGER, name="W")
H=model.addVars(T, obj=h,vtype=GRB.INTEGER, name="H")
F=model.addVars(T, obj=f,vtype=GRB.INTEGER, name="F")
P=model.addVars(T, obj=p,vtype=GRB.INTEGER, name="P")
I=model.addVars(T, obj=i,vtype=GRB.INTEGER, name="I")
S=model.addVars(T, obj=s,vtype=GRB.INTEGER, name="S")
C=model.addVars(T, obj=c,vtype=GRB.INTEGER, name="C")
O=model.addVars(T, obj=o,vtype=GRB.INTEGER, name="O")
```

The Costs

The demand

Example of inserting directly from python (not reading in excel)

* Note since single commodity you don't have to add the name of the commodity to the dictionary

Building and solving the model with Gurobi/Python

Constraints for every period if the period is greater than 1. Why? Since

Note that you do not have to change any of this

```
#Constraints

#Constraints (1)
model.addConstr((W[1]==b+H[1]-F[1]), "c1a")
model.addConstrs((W[t]==W[t-1]+H[t]-F[t] for t in T if t>1),"c1b")

#Constraints (2)
model.addConstrs((k*P[t]<=n*W[t]+O[t] for t in T),"c2")

#Constraints (3)
model.addConstr((a+P[1]+C[1])-e==d[1]+I[1]-S[1],"c3a")
model.addConstrs((I[t-1]+P[t]+C[t]-S[t-1]==d[t]+I[t]-S[t] for t in T if t>1),"c3b")

#Constraints (4)
model.addConstrs((O[t]<=m*W[t] for t in T),"c")

#model.addConstrs((),"c")

model.update()

#Objective function
model.setParam('OutputFlag',0)
model.optimize()

#Print solution
if model.status==GRB.Status.OPTIMAL:
    print('Obj: %g' % model.objVal)
    for v in model.getVars():
        print('%s %g' % (v.varName, v.x))
```

Example – Red Tomato Tools

Total cost: $284,600

Period, t	No. Hired, H_t	No. Laid Off, L_t	Workforce Size, W_t	Overtime, O_t	Inventory, I_t	Stockout, S_t	Subcontract, C_t	Total Production, P_t	Demand, D_t
0	0	0	80	0	0	0	0		
1	0	65	15	0	0	0	0	600	1,600
2	0	15	0	0	0	3000	0	0	3,000
3	0	0	0	0	0	6200	0	0	3,200
4	0	0	0	0	0	10000	0	0	3,800
5	0	0	0	0	0	12200	0	0	2,200
6	0	0	0	0	0	14400	0	0	2,200

This model is telling you that the business model is not good and you should not do anything

Example – Red Tomato Tools

Total cost: $422,660

Period, t	No. Hired, H_t	No. Laid Off, L_t	Workforce Size, W_t	Overtime, O_t	Inventory, I_t	Stockout, S_t	Subcontract, C_t	Total Production, P_t	Demand, D_t
0	0	0	80	0	1,000	0	0		
1	0	16	64	0	1,960	0	0	2,583	1,600
2	0	0	64	0	1,520	0	0	2,583	3,000
3	0	0	64	0	880	0	0	2,583	3,200
4	0	0	64	0	0	220	140	2,583	3,800
5	0	0	64	0	140	0	0	2,583	2,200
6	0	0	64	0	500	0	0	2,583	2,200

Including additional constraints:

- Final Backlog =0
- Final inventory = 500

Constraints:

Supply[finalIndex] == 0
Inventory[finalIndex] == 500

Input Data - Excel

	A	B	C	D	E	F	G	H
1	Parameters (not indexed)							
2	number of periods (cardinality of Set T)	6						
3	number of commodities (cardinality of Set L)	1						
4	Regular time cost (Parameter: w)	4						
5	Overtime cost (Parameter: o)	6						
6	Hiring and training costs (Parameter: h)	300						
7	Layoff costs (Parameter: f)	500						
8	Initial workforce (Parameter: b)	80						
9								
10								
11		Periods (Set: T)						
12	Parameters (indexed in T)	1	2	3	4	5	6	
13	Number of regular working hours per worker (Parameter: n)	160	160	160	160	160	160	
14	Maximum number of overtime hours per worker (Parameter:	10	10	10	10	10	10	
15								
16								
17								
18		Commodities (Set: L)						
19	Parameters (indexed in L)	Product A						
20	Marginal costs of stockout backlog (Parameter: s)	5						
21	Labour hours required per unit produced (Parameter: k)	4						
22	Subcontracting costs (Parameter: c)	30						
23	Initial inventory (Parameter: a)	1000						
24	Initial backlog (Parameter: e)	0						
25								
26								
27								
28								
29			Periods (Set: T)					
30	Parameters (indexed in L and T)	Commodities (Set: L)	1	2	3	4	5	6
31	Demand (Parameter: d)	Product A	1600	3000	3200	3800	2200	2200
32	Unit production costs (Parameter: p)	Product A	10	10	10	10	10	10
33	Inventory holding costs (Parameter: i)	Product A	2	2	2	2	2	2
34								
35								
36								
37								
38								
39								

AggregatePlanning_SC_Input | AggregatePlanning_MC_Input

Yellow Cell

Green Cell

Blue Cell

Reading Input Data from Excel

```python
from gurobipy import *
import openpyxl as opxl
#import matplotlib.pyplot as plt

#Provide Excel file and sheet name
fileXLS="AggregatePlanningXLS_Example.xlsx"
sheetXLS="AggregatePlanning_SC_Input"

doc = opxl.load_workbook(fileXLS)

#Read number of periods and commodities (nT and nL, respectively)
nT=doc[sheetXLS].cell(row = 2, column = 2).value
nL=doc[sheetXLS].cell(row = 3, column = 2).value

#Read parameters that are not indexed (w, o, h, f, b)
w=doc[sheetXLS].cell(row = 4, column = 2).value
o=doc[sheetXLS].cell(row = 5, column = 2).value
h=doc[sheetXLS].cell(row = 6, column = 2).value
f=doc[sheetXLS].cell(row = 7, column = 2).value
b=doc[sheetXLS].cell(row = 8, column = 2).value

#Read parameters that are indexed in T

#provide col and row for yellow cell
yrow=12
ycol=1

#Read set T and parameters n and m
T=[doc[sheetXLS].cell(row = yrow, column = ycol + col ).value for col in range(1,nT+1)]
n = {T[col-1]:doc[sheetXLS].cell(row = yrow+1, column = ycol + col).value for col in range(1,nT+1)}
m = {T[col-1]:doc[sheetXLS].cell(row = yrow+2, column = ycol + col).value for col in range(1,nT+1)}

```

Reading Input Data from Excel

```
#Read parameters that are indexed in L

#provide col and row for green cell
grow=19
gcol=1

#Read set L and parameters s, k, c, a, e, h, and f
L=[doc[sheetXLS].cell(row = grow, column = gcol + col ).value for col in range(1,nL+1)]
s = {L[col-1]:doc[sheetXLS].cell(row = grow+1, column = gcol+col).value for col in range(1,nL+1)}
k = {L[col-1]:doc[sheetXLS].cell(row = grow+2, column = gcol+col).value for col in range(1,nL+1)}
c = {L[col-1]:doc[sheetXLS].cell(row = grow+3, column = gcol+col).value for col in range(1,nL+1)}
a = {L[col-1]:doc[sheetXLS].cell(row = grow+4, column = gcol+col).value for col in range(1,nL+1)}
e = {L[col-1]:doc[sheetXLS].cell(row = grow+5, column = gcol+col).value for col in range(1,nL+1)}

#Read parameters that are indexed in T and L

#provide col and row for blue cell
brow=30
bcol=1

#Read parameters d, p, and i
d = {(L[rowAux-1],T[colAux-1]):doc[sheetXLS].cell(row = brow+1, column = bcol+1+colAux).value for rowAux in range(1,nL+1) for co
lAux in range(1,nT+1)}
p = {(L[rowAux-1],T[colAux-1]):doc[sheetXLS].cell(row = brow+nL+1, column = bcol+1+colAux).value for rowAux in range(1,nL+1) for
colAux in range(1,nT+1)}
i = {(L[rowAux-1],T[colAux-1]):doc[sheetXLS].cell(row = brow+2*nL+1, column = bcol+1+colAux).value for rowAux in range(1,nL+1) f
or colAux in range(1,nT+1)}
```

Building optimization model (creating variables using Dictionaries)

You can use "model.addvars", or you can create a dictionary of individual variables (using "model.addVar") where the keys are the desired indices

```
#### OPTIMIZATION MODEL ####################################################

my_model = Model('AggregatePlanning')
my_model.setParam(GRB.Param.OutputFlag, 0)

#Decision variables
W = {t:my_model.addVar(vtype=GRB.INTEGER,name="W["+str(t)+"]") for t in T}
O = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="O["+str(t)+"]") for t in T}
H = {t:my_model.addVar(vtype=GRB.INTEGER,name="H["+str(t)+"]") for t in T}
F = {t:my_model.addVar(vtype=GRB.INTEGER,name="F["+str(t)+"]") for t in T}
P = {(l,t):my_model.addVar(vtype=GRB.CONTINUOUS,name="P["+str(l)+","+str(t)+"]") for t in T for l in L}
I = {(l,t):my_model.addVar(vtype=GRB.CONTINUOUS,name="I["+str(l)+","+str(t)+"]") for t in T for l in L}
S = {(l,t):my_model.addVar(vtype=GRB.CONTINUOUS,name="S["+str(l)+","+str(t)+"]") for t in T for l in L}
C = {(l,t):my_model.addVar(vtype=GRB.CONTINUOUS,name="C["+str(l)+","+str(t)+"]") for t in T for l in L}

#Objective Function

#Regular-time labor cost
RTLC = quicksum(n[t]*w*W[t] for t in T)
#Overtime labor cost
OTLC = quicksum(o*O[t] for t in T)
#Cost of hiring
HC = quicksum(h*H[t] for t in T)
#Cost of layoffs
FC = quicksum(f*F[t] for t in T)
#Cost of holding inventory
HIC = quicksum(i[l,t]*I[l,t] for t in T for l in L)
#Cost of stocking out
CSO = quicksum(s[l]*S[l,t] for t in T for l in L)
#Production cost
PC = quicksum(p[l,t]*P[l,t] for t in T for l in L)
#Subcontracting cost
SC = quicksum(c[l]*C[l,t] for t in T for l in L)


FO = (RTLC+ OTLC+ HC+ FC+ HIC+ CSO+ PC+ SC)

my_model.setObjective(FO,GRB.MINIMIZE)
```

Creating the associated constraints and solving the model

```
##Constraints

#Workforce, hiring, and layoff constraints
my_model.addConstr(W[1]==b+H[1]-F[1])
my_model.addConstrs(W[t]==W[t-1]+H[t]-F[t] for t in T if t is not T[0])

#Capacity constraints
my_model.addConstrs(sum(k[l]*P[l,t] for l in L)<=n[t]*W[t]+O[t] for t in T)

#Inventory balance constraints
my_model.addConstrs(a[l]+P[l,1]+C[l,1]-e[l]==d[l,1]+I[l,1]-S[l,1] for l in L)
my_model.addConstrs(I[l,t-1]+P[l,t]+C[l,t]-S[l,t-1]==d[l,t]+I[l,t]-S[l,t] for l in L for t in T if t is not T[0])

#Overtime constraints
my_model.addConstrs(O[t]<=m[t]*W[t] for t in T)

#Additional constraints (final backlog=0 and final inventory=500)

my_model.addConstrs(sum(S[l,t] for l in L)==0 for t in T[-1:])
my_model.addConstrs(sum(I[l,t] for l in L)==500 for t in T[-1:])

my_model.update()
my_model.optimize()
```

This is good to guarantee the same data type as the period

list[-1] the last value in the list.

In context, it means until the last value in the list

Print solutions (to console and to Excel)

```
###PRINT SOLUTION######################################################################
### Print solution in Console ##################################################

if my_model.status==GRB.Status.OPTIMAL:
    print('Obj: %g' % my_model.objVal)
    for v in my_model.getVars():
        print('%s %g' % (v.varName, v.x))

#Print solution to EXCEL ####################################################
import xlwt
from xlwt import Workbook

# Workbook is created
wb = Workbook()

# add_sheet is used to create sheet.
sheet1 = wb.add_sheet('AggregatePlanningXLS_Output')

#print titles
sheet1.write(0, 0, 'Period')
sheet1.write(0, 1, 'H')
sheet1.write(0, 2, 'F')
sheet1.write(0, 3, 'W')
sheet1.write(0, 4, 'O')
for lAux in range(1,nL+1):
    sheet1.write(0, 4*lAux+1, 'I['+ str(L[lAux-1])+']')
    sheet1.write(0, 4*lAux+2, 'S['+ str(L[lAux-1])+']')
    sheet1.write(0, 4*lAux+3, 'C['+ str(L[lAux-1])+']')
    sheet1.write(0, 4*lAux+4, 'P['+ str(L[lAux-1])+']')

#print values
row=1
for t in T:
    sheet1.write(row, 0, t)
    sheet1.write(row, 1, H[t].x)
    sheet1.write(row, 2, F[t].x)
    sheet1.write(row, 3, W[t].x)
    sheet1.write(row, 4, O[t].x)
    for lAux in range(1,nL+1):
        sheet1.write(row, 4*lAux+1, I[L[lAux-1],t].x)
        sheet1.write(row, 4*lAux+2, S[L[lAux-1],t].x)
        sheet1.write(row, 4*lAux+3, C[L[lAux-1],t].x)
        sheet1.write(row, 4*lAux+4, P[L[lAux-1],t].x)
    row+=1

#Save Excel file
wb.save("AggregatePlanningXLS_Solution.xls")
```

Solution in Excel

	A	B	C	D	E	F	G	H	I
1	Period	H	F	W	O	I[Product A]	S[Product A]	C[Product A]	P[Product A]
2	1	0	16	64	0	1960	0	0	2560
3	2	0	0	64	0	1520	0	0	2560
4	3	0	0	64	0	880	0	0	2560
5	4	0	0	64	0	0	220	140	2560
6	5	0	0	64	0	140	0	0	2560
7	6	0	0	64	0	500	0	0	2560
8									
9									
10									

Example – Multiple commodities

Parameters (not indexed)	
number of periods (cardinality of Set T)	6
number of commodities (cardinality of Set L)	2
Regular time cost (Parameter: w)	4
Overtime cost (Parameter: o)	6
Hiring and training costs (Parameter: h)	6
Layoff costs (Parameter: f)	500
Initial workforce (Parameter: b)	80

	Periods (Set: T)					
Parameters (indexed in T)	**1**	**2**	**3**	**4**	**5**	**6**
Number of regular working hours per worker (Parameter: n)	160	160	160	160	160	160
Maximum number of overtime hours per worker (Parameter:	10	10	10	10	10	10

	Commodities (Set: L)	
Parameters (indexed in L)	**Product A**	**Product B**
Marginal costs of stockout backlog (Parameter: s)	5	7
Labour hours required per unit produced (Parameter: k)	4	5
Subcontracting costs (Parameter: c)	30	40
Initial inventory (Parameter: a)	1000	500
Initial backlog (Parameter: e)	0	50

		Periods (Set: T)					
Parameters (indexed in L and T)	**Commodities (Set: L)**	**1**	**2**	**3**	**4**	**5**	**6**
Demand (Parameter: d)	Product A	1600	3000	3200	3800	2200	2200
	Product B	2600	3000	2500	3500	2700	2100
Unit production costs (Parameter: p)	Product A	10	10	10	10	10	10
	Product B	20	20	25	25	25	25
Inventory holding costs (Parameter: i)	Product A	2	2	2	2	2	2
	Product B	1	1	1	3	3	3

AggregatePlanning_SC_Input | AggregatePlanning_MC_Input

Change data here but do not need to change in python

Solution in console and Excel

```
Obj: 890054
W[1] 83
W[2] 158
W[3] 158
W[4] 158
W[5] 158
W[6] 158
O[1] 0
O[2] 0
O[3] 0
O[4] 0
O[5] 0
O[6] 0
H[1] 3
H[2] 75
H[3] 0
H[4] 0
H[5] 0
H[6] 0
F[1] -0
F[2] -0
F[3] -0
F[4] -0
F[5] 0
F[6] -0
P[Product A,1] 600
P[Product B,1] 2176
P[Product A,2] 3000
P[Product B,2] 2656
P[Product A,3] 3172.5
P[Product B,3] 2518
P[Product A,4] 1570
P[Product B,4] 3800
P[Product A,5] 3570
P[Product B,5] 2200
P[Product A,6] 3570
P[Product B,6] 2200
```

```
I[Product A,1] 0
I[Product B,1] 1026
I[Product A,2] 0
I[Product B,2] 682
I[Product A,3] 0
I[Product B,3] 0
I[Product A,4] 0
I[Product B,4] 0
I[Product A,5] 0
I[Product B,5] 0
I[Product A,6] 500
I[Product B,6] 0
S[Product A,1] 0
S[Product B,1] 0
S[Product A,2] 0
S[Product B,2] 0
S[Product A,3] 10
S[Product B,3] 0
S[Product A,4] 2240
S[Product B,4] 0
S[Product A,5] 870
S[Product B,5] 0
S[Product A,6] 0
S[Product B,6] 0
C[Product A,1] 0
C[Product B,1] 0
C[Product A,2] 0
C[Product B,2] 0
C[Product A,3] 17.5
C[Product B,3] 0
C[Product A,4] 0
C[Product B,4] 0
C[Product A,5] 4.54747e-13
C[Product B,5] 0
C[Product A,6] 0
C[Product B,6] 0
```

Period	H	F	W	O	I[Product A]	S[Product A]	C[Product A]	P[Product A]	I[Product B]	S[Product B]	C[Product B]	P[Product B]
1	3	0	83	0	0	0	0	600	1026	0	0	2176
2	75	0	158	0	0	0	0	3000	682	0	0	2656
3	0	0	158	0	0	10	17.5	3172.5	0	0	0	2518
4	0	0	158	0	0	2240	0	1570	0	0	0	3800
5	0	0	158	0	0	870	4.54747E-13	3570	0	0	0	2200
6	0	0	158	0	500	0	0	3570	0	0	0	2200

Note that some variables are fractional since in this example we did not enforce integrality on all variables. Depending on the context, you may prefer continuous or integer variables

THANK YOU

QUESTIONS?

Andrés D. González | andres.gonzalez@ou.edu