# ISE 4623/5023: Deterministic Systems Models / Systems Optimization

**University of Oklahoma**
**School of Industrial and Systems Engineering**
**Fall 2021**

**Individual assignment 7 (100 points + 30 points for graduate students/extra credit)**

***NOTE: For all problems, you need to upload a PDF file of the solution, along with support files of any software used (Excel, Gurobi/Python, etc).***

---

***PLEDGE:***

***"On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise."***

***Name:*** _____ ***Signature:*** _____

***Student ID:*** _____ ***Date:*** _____

---

**Problem 1 (50 points + 10 extra credit)**

The U.S. Government has a budget of 200 million USD to carry out road infrastructure works (Construction Projects), to generate development in the country. They have shared with you the file " ConstructionProjects.xlsx", which contains information corresponding to the cost (in millions of USD) of execution and the number of jobs (in thousands of people) generated, for each construction project belonging to the set of road infrastructure works that can be executed. The objective is to generate as many jobs as possible through the works to be executed.

a. (25 points) Formulate a mathematical optimization model that allows you to determine which construction projects
should be carried out. Define clearly and rigorously:

I. Sets
$S$: Set of construction projects

II. Parameters
$c_i$: Cost of executing project $i \in S$.
$b_i$: Jobs generated for executing project $i \in S$.
$k$: Available budget.

III. Decision Variables
$x_i$: 1 if construction project $i \in S$ is executed, 0 otherwise.

IV. Objective Function

$$\max \sum_{i \in S} x_i \cdot b_i$$

V. Constraints

s.t.,

$$\sum_{i \in S} x_i \cdot c_i \leq k$$

$$x_i \in \{0,1\} \; \forall \; i \in S$$

b. (25 points) Solve the problem using Gurobi. What is the optimal solution to the problem? Present in a friendly way the value of the decision variables and the objective function at the optimum (if any), and any other result that you might consider relevant. Interpret and conclude regarding your solution in terms of the context of the problem.

```
from gurobipy import *

#Abra el archivo excel
import openpyxl as opxl

#Load the parameters from Excel file
doc = opxl.load_workbook("ConstructionProjects.xlsx")

#Sets
CP = []
row = 1
while doc["ConstructionProjects"].cell(row = row+1, column = 2).value:
        cp = doc["ConstructionProjects"].cell(row = row+1, column = 1).value
        CP.append(cp)
        row += 1


#Parameters
ci = {}
row = 1
while doc["ConstructionProjects"].cell(row = row+1, column = 2).value:
        cp = doc["ConstructionProjects"].cell(row = row+1, column = 1).value
        cost_cp = doc["ConstructionProjects"].cell(row = row+1, column = 2).value
        ci[cp] = cost_cp
        row += 1
```

```
bi = {}
row = 1
while doc["ConstructionProjects"].cell(row = row+1, column = 3).value:
        cp = doc["ConstructionProjects"].cell(row = row+1, column = 1).value
        jobs_cp = doc["ConstructionProjects"].cell(row = row+1, column = 3).value
        bi[cp] = jobs_cp
        row += 1

k = 200

#Model
m = Model('Knapsack')
m.setParam(GRB.Param.OutputFlag, 0)
#Variables
x = {i:m.addVar(vtype=GRB.BINARY,name="x_"+str(i)) for i in CP}

m.addConstr(quicksum(x[i]*ci[i] for i in CP) <= k)

FO = quicksum(x[i]*bi[i] for i in CP)

m.setObjective(FO,GRB.MAXIMIZE)
m.update()
m.optimize()

print("\n\tThe optimal value for the objective function is: "+str(m.objVal))
print("\tThe variables that take vale are:")
for i in CP:
    if x[i].x > 0.5 :
        print("\t\tx"+str(i)+" = "+str(x[i].x))
```

Academic license - for non-commercial use only - expires 2022-08-12
Using license file C:\Users\samue\gurobi.lic

The optimal value for the objective function is: 302.09999999999997
The variables that take vale are:
x0 = 1.0
x1 = 1.0
x2 = 1.0
x3 = -0.0
x4 = -0.0
x5 = 1.0
x6 = -0.0
x7 = -0.0
x8 = 1.0
x9 = 1.0
x10 = 0.0
x11 = -0.0
x12 = 0.0
x13 = -0.0
x14 = 1.0
x15 = 1.0
x16 = 1.0
x17 = 1.0
x18 = 1.0
x19 = 1.0
x20 = -0.0
x21 = -0.0
x22 = 1.0
x23 = 1.0

x24 = -0.0
x25 = 1.0
x26 = -0.0
x27 = -0.0
x28 = 1.0
x29 = 1.0
x30 = -0.0
x31 = 1.0
x32 = 1.0
x33 = -0.0
x34 = 1.0
x35 = 1.0
x36 = 1.0
x37 = 1.0
x38 = 1.0
x39 = 1.0
x40 = 1.0
x41 = 1.0
x42 = -0.0
x43 = 1.0
x44 = -0.0
x45 = 1.0
x46 = -0.0
x47 = 1.0
x48 = 0.0
x49 = -0.0
x50 = 0.0
x51 = -0.0
x52 = 1.0
x53 = 1.0
x54 = 1.0
x55 = -0.0
x56 = -0.0
x57 = 1.0
x58 = 1.0
x59 = 1.0
x60 = -0.0
x61 = -0.0
x62 = 1.0
x63 = 1.0
x64 = -0.0
x65 = 1.0
x66 = 1.0
x67 = 0.0
x68 = 1.0
x69 = -0.0
x70 = 1.0
x71 = -0.0
x72 = 1.0
x73 = 1.0
x74 = -0.0
x75 = 1.0
x76 = 1.0
x77 = 1.0
x78 = 1.0
x79 = -0.0
x80 = 1.0
x81 = -0.0

x82 = 1.0
x83 = 1.0
x84 = -0.0
x85 = -0.0
x86 = 1.0
x87 = 1.0
x88 = 1.0
x89 = 1.0
x90 = 1.0
x91 = 1.0
x92 = 1.0
x93 = 1.0
x94 = 1.0
x95 = 1.0
x96 = -0.0
x97 = -0.0
x98 = 1.0
x99 = -0.0
x100 = -0.0
x101 = -0.0
x102 = 1.0
x103 = 1.0
x104 = 1.0
x105 = 1.0
x106 = -0.0
x107 = 1.0
x108 = -0.0
x109 = 1.0
x110 = -0.0
x111 = -0.0
x112 = -0.0
x113 = -0.0
x114 = 1.0
x115 = 1.0
x116 = 1.0
x117 = 1.0
x118 = 1.0
x119 = 1.0
x120 = -0.0
x121 = -0.0
x122 = 1.0
x123 = 1.0
x124 = 0.0
x125 = 1.0
x126 = 1.0
x127 = 1.0
x128 = -0.0
x129 = 1.0
x130 = -0.0
x131 = -0.0
x132 = -0.0
x133 = 1.0
x134 = 1.0
x135 = -0.0
x136 = 1.0
x137 = 1.0
x138 = 1.0
x139 = -0.0

x140 = 1.0
x141 = 1.0
x142 = 1.0
x143 = -0.0
x144 = 1.0
x145 = 0.0
x146 = 1.0
x147 = 1.0
x148 = -0.0
x149 = -0.0
x150 = 1.0
x151 = 1.0
x152 = 0.0
x153 = 1.0
x154 = 1.0
x155 = 1.0
x156 = 1.0
x157 = -0.0
x158 = 1.0
x159 = 1.0
x160 = 1.0
x161 = 1.0
x162 = 1.0
x163 = -0.0
x164 = 1.0
x165 = 1.0
x166 = 0.0
x167 = 1.0
x168 = 0.0
x169 = -0.0
x170 = 1.0
x171 = -0.0
x172 = 1.0
x173 = -0.0
x174 = 0.0
x175 = -0.0
x176 = 1.0
x177 = 1.0
x178 = 1.0
x179 = 1.0
x180 = 1.0
x181 = 1.0
x182 = 1.0
x183 = 1.0
x184 = 1.0
x185 = 1.0
x186 = 1.0
x187 = 1.0
x188 = -0.0
x189 = 1.0
x190 = 1.0
x191 = -0.0
x192 = 1.0
x193 = 1.0
x194 = 1.0
x195 = -0.0
x196 = -0.0
x197 = 1.0

x198 = -0.0
x199 = -0.0
x200 = -0.0
x201 = 1.0
x202 = -0.0
x203 = -0.0
x204 = 1.0
x205 = 1.0
x206 = 1.0
x207 = 1.0
x208 = 1.0
x209 = 1.0
x210 = -0.0
x211 = 1.0
x212 = 1.0
x213 = 1.0
x214 = -0.0
x215 = -0.0
x216 = 1.0
x217 = -0.0
x218 = 1.0
x219 = 0.0
x220 = 1.0
x221 = 1.0
x222 = -0.0
x223 = -0.0
x224 = -0.0
x225 = 0.0
x226 = 1.0
x227 = -0.0
x228 = 1.0
x229 = 1.0
x230 = 1.0
x231 = 1.0
x232 = 1.0
x233 = 1.0
x234 = -0.0
x235 = 1.0
x236 = 1.0
x237 = 1.0
x238 = 1.0
x239 = 1.0
x240 = -0.0
x241 = -0.0
x242 = 1.0
x243 = 1.0
x244 = 1.0
x245 = 1.0
x246 = 1.0
x247 = -0.0
x248 = -0.0
x249 = 1.0

c. (10 points for graduate students/extra credit) How would the solution change if now there is a requirement to execute construction project #10 if construction projects #8 and #9 are executed (i.e., #10 can be executed without #8 and #9 being executed but #10 MUST be executed if both projects #8 and #9 are executed).

New constraint:

$$x_{10} \geq x_8 + x_9 - 1$$

```
m.addConstr(x[8]+x[7]-1 <= x[9])
```

Note: Indexes start at 0.

The optimal value for the objective function is: 302.09999999999997

The variables that take vale are:

x0 = 1.0
x1 = 1.0
x2 = 1.0
x3 = 0.0
x4 = 0.0
x5 = 1.0
x6 = 0.0
x7 = 0.0
x8 = 1.0
x9 = 1.0
x10 = 0.0
x11 = 0.0
x12 = 0.0
x13 = 0.0
x14 = 1.0
x15 = 1.0
x16 = 1.0
x17 = 1.0
x18 = 1.0
x19 = 1.0
x20 = 0.0
x21 = 0.0
x22 = 1.0
x23 = 1.0
x24 = 0.0
x25 = 1.0
x26 = 0.0
x27 = 0.0
x28 = 1.0
x29 = 1.0
x30 = 0.0
x31 = 1.0
x32 = 1.0
x33 = 0.0
x34 = 1.0
x35 = 1.0
x36 = 1.0
x37 = 1.0
x38 = 1.0
x39 = 1.0
x40 = 1.0
x41 = 1.0
x42 = 0.0

x43 = 1.0
x44 = 0.0
x45 = 1.0
x46 = 0.0
x47 = 1.0
x48 = 0.0
x49 = 0.0
x50 = 0.0
x51 = 0.0
x52 = 1.0
x53 = 1.0
x54 = 1.0
x55 = 0.0
x56 = 0.0
x57 = 1.0
x58 = 1.0
x59 = 1.0
x60 = 0.0
x61 = 0.0
x62 = 1.0
x63 = 1.0
x64 = 0.0
x65 = 1.0
x66 = 1.0
x67 = 0.0
x68 = 1.0
x69 = 0.0
x70 = 1.0
x71 = 0.0
x72 = 1.0
x73 = 1.0
x74 = 0.0
x75 = 1.0
x76 = 1.0
x77 = 1.0
x78 = 1.0
x79 = 0.0
x80 = 1.0
x81 = 0.0
x82 = 1.0
x83 = 1.0
x84 = 0.0
x85 = 0.0
x86 = 1.0
x87 = 1.0
x88 = 1.0
x89 = 1.0
x90 = 1.0
x91 = 1.0
x92 = 1.0
x93 = 1.0
x94 = 1.0
x95 = 1.0
x96 = 0.0
x97 = 0.0
x98 = 1.0
x99 = 0.0
x100 = 0.0

x101 = 0.0
x102 = 1.0
x103 = 1.0
x104 = 1.0
x105 = 1.0
x106 = 0.0
x107 = 1.0
x108 = 0.0
x109 = 1.0
x110 = 0.0
x111 = 0.0
x112 = 0.0
x113 = 0.0
x114 = 1.0
x115 = 1.0
x116 = 1.0
x117 = 1.0
x118 = 1.0
x119 = 1.0
x120 = 0.0
x121 = 0.0
x122 = 1.0
x123 = 1.0
x124 = 0.0
x125 = 1.0
x126 = 1.0
x127 = 1.0
x128 = 0.0
x129 = 1.0
x130 = 0.0
x131 = 0.0
x132 = 0.0
x133 = 1.0
x134 = 1.0
x135 = 0.0
x136 = 1.0
x137 = 1.0
x138 = 1.0
x139 = 0.0
x140 = 1.0
x141 = 1.0
x142 = 1.0
x143 = 0.0
x144 = 1.0
x145 = 0.0
x146 = 1.0
x147 = 1.0
x148 = 0.0
x149 = 0.0
x150 = 1.0
x151 = 1.0
x152 = 0.0
x153 = 1.0
x154 = 1.0
x155 = 1.0
x156 = 1.0
x157 = 0.0
x158 = 1.0

x159 = 1.0
x160 = 1.0
x161 = 1.0
x162 = 1.0
x163 = 0.0
x164 = 1.0
x165 = 1.0
x166 = 0.0
x167 = 1.0
x168 = 0.0
x169 = 0.0
x170 = 1.0
x171 = 0.0
x172 = 1.0
x173 = 0.0
x174 = 0.0
x175 = 0.0
x176 = 1.0
x177 = 1.0
x178 = 1.0
x179 = 1.0
x180 = 1.0
x181 = 1.0
x182 = 1.0
x183 = 1.0
x184 = 1.0
x185 = 1.0
x186 = 1.0
x187 = 1.0
x188 = 0.0
x189 = 1.0
x190 = 1.0
x191 = 0.0
x192 = 1.0
x193 = 1.0
x194 = 1.0
x195 = 0.0
x196 = 0.0
x197 = 1.0
x198 = 0.0
x199 = 0.0
x200 = 0.0
x201 = 1.0
x202 = 0.0
x203 = 0.0
x204 = 1.0
x205 = 1.0
x206 = 1.0
x207 = 1.0
x208 = 1.0
x209 = 1.0
x210 = 0.0
x211 = 1.0
x212 = 1.0
x213 = 1.0
x214 = 0.0
x215 = 0.0
x216 = 1.0

x217 = 0.0
x218 = 1.0
x219 = 0.0
x220 = 1.0
x221 = 1.0
x222 = 0.0
x223 = 0.0
x224 = 0.0
x225 = 0.0
x226 = 1.0
x227 = 0.0
x228 = 1.0
x229 = 1.0
x230 = 1.0
x231 = 1.0
x232 = 1.0
x233 = 1.0
x234 = 0.0
x235 = 1.0
x236 = 1.0
x237 = 1.0
x238 = 1.0
x239 = 1.0
x240 = 0.0
x241 = 0.0
x242 = 1.0
x243 = 1.0
x244 = 1.0
x245 = 1.0
x246 = 1.0
x247 = 0.0
x248 = 0.0
x249 = 1.0

**Problem 2 (50 points + 20 extra credit)**

OptiCoffee is a company dedicated to the production and distribution of coffee in the city of Norman, OK. The company's production area has contacted you to plan the production strategy of tons of coffee for the next 12 months. The Production area must decide how many tons of coffee should be produced, both by themselves and/or by subcontracting a third party, to satisfy the demand (tons) at the end of each month. Currently, OptiCoffee has an inventory of 100 tons of coffee stored in its warehouse and a backlog of 80 tons of coffee. At the end of each month, the coffee that has not been used will be stored in inventory and can be used to meet the demand or the backlogs for the following and previous months, respectively.

Additionally, OptiCoffee has a current workforce that consists of 7 employees, and the company can decide to hire or lay off employees each month. The tons of coffee produced require a certain amount of labor hours from the employees to be produced and the employees have a regular work time with the possibility of overworking without exceeding a certain amount of time (e.g., For month 1, assuming no hiring or layoffs, the maximum number of hours worked (the product between the tons produced and the hours required hours required to produce them) cannot exceed month's 1 maximum time capacity (the product between the number of regular of working hours and the workforce, plus the number of extra hours worked for that month).

Given this information, OptiCoffee's production area wants to minimize their total costs throughout the planning horizon (i.e., 12 months). All the costs associated to the production process (e.g., production, subcontracting, holding

inventory, hiring employees, lay off costs, regular/overtime work costs, etc.), as well as other relevant information respecting the workforce can be found in the “OptiCoffee.xlsx” file.

a. (5 points) Solve the model using Gurobi. What is the optimal value of the objective function?

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

doc = opxl.load_workbook("OptiCoffee.xlsx")

M = range(1,13)

#demand
d = {t:doc["Information"].cell(row = 4, column = t+1).value for t in M}
#production costs
p = {t:doc["Information"].cell(row = 5, column = t+1).value for t in M}
#Holding inventory costs
i = {t:doc["Information"].cell(row = 6, column = t+1).value for t in M}
#Number of regular working hours
n = {t:doc["Information"].cell(row = 7, column = t+1).value for t in M}
#Maximum number of overtime hours
m = {t:doc["Information"].cell(row = 8, column = t+1).value for t in M}
#Marginal cost of stockout/backlog
s = doc["Information"].cell(row = 9, column = 2).value
#Hiring and training costs
h = doc["Information"].cell(row = 10, column = 2).value
#Layoff cost
f = doc["Information"].cell(row = 11, column = 2).value
#Labor hours required per ton produced
k = doc["Information"].cell(row = 12, column = 2).value
#Regular time cost
w = doc["Information"].cell(row = 13, column = 2).value
```

```
#Overtime cost
o = doc["Information"].cell(row = 14, column = 2).value
#Cost of subcontracting a ton
c = doc["Information"].cell(row = 15, column = 2).value
#Initial inventory
a = doc["Information"].cell(row = 16, column = 2).value
#Initial workforce
b = doc["Information"].cell(row = 17, column = 2).value
#Initial backlog
e = doc["Information"].cell(row = 18, column = 2).value
```

```
my_model = Model('OptiCoffee')
my_model.setParam(GRB.Param.OutputFlag, 0)
#Workforce size
W = {t:my_model.addVar(vtype=GRB.INTEGER,name="W_"+str(t)) for t in M}
#Employees hired
H = {t:my_model.addVar(vtype=GRB.INTEGER,name="H_"+str(t)) for t in M}
#Employees laid off
F = {t:my_model.addVar(vtype=GRB.INTEGER,name="F_"+str(t)) for t in M}
#Production
P = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="P_"+str(t)) for t in M}
#Inventory at the end of the month
I = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="I_"+str(t)) for t in M}
#Tons stocked at the end of the month
S = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="S_"+str(t)) for t in M}
#Tons subcontracted at the end of the month
C = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="C_"+str(t)) for t in M}
#Number of overtime hours worked
O = {t:my_model.addVar(vtype=GRB.CONTINUOUS,name="O_"+str(t)) for t in M}

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


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


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

```
#Workforce, hiring, and layoff constraints
my_model.addConstr(W[1]==b+H[1]-F[1])
for t in M:
    if t > 1:
        my_model.addConstr(W[t]==W[t-1]+H[t]-F[t])

#Capacity constraints
for t in M:
    my_model.addConstr(k*P[t]<=n[t]*W[t]+O[t])

#Inventory balance constraints
my_model.addConstr(a+P[1]+C[1]-e==d[1]+I[1]-S[1])
for t in M:
    if t > 1:
        my_model.addConstr(I[t-1]+P[t]+C[t]-S[t-1]==d[t]+I[t]-S[t])

#Overtime constraints
for t in M:
    my_model.addConstr(O[t]<=m[t]*W[t])

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

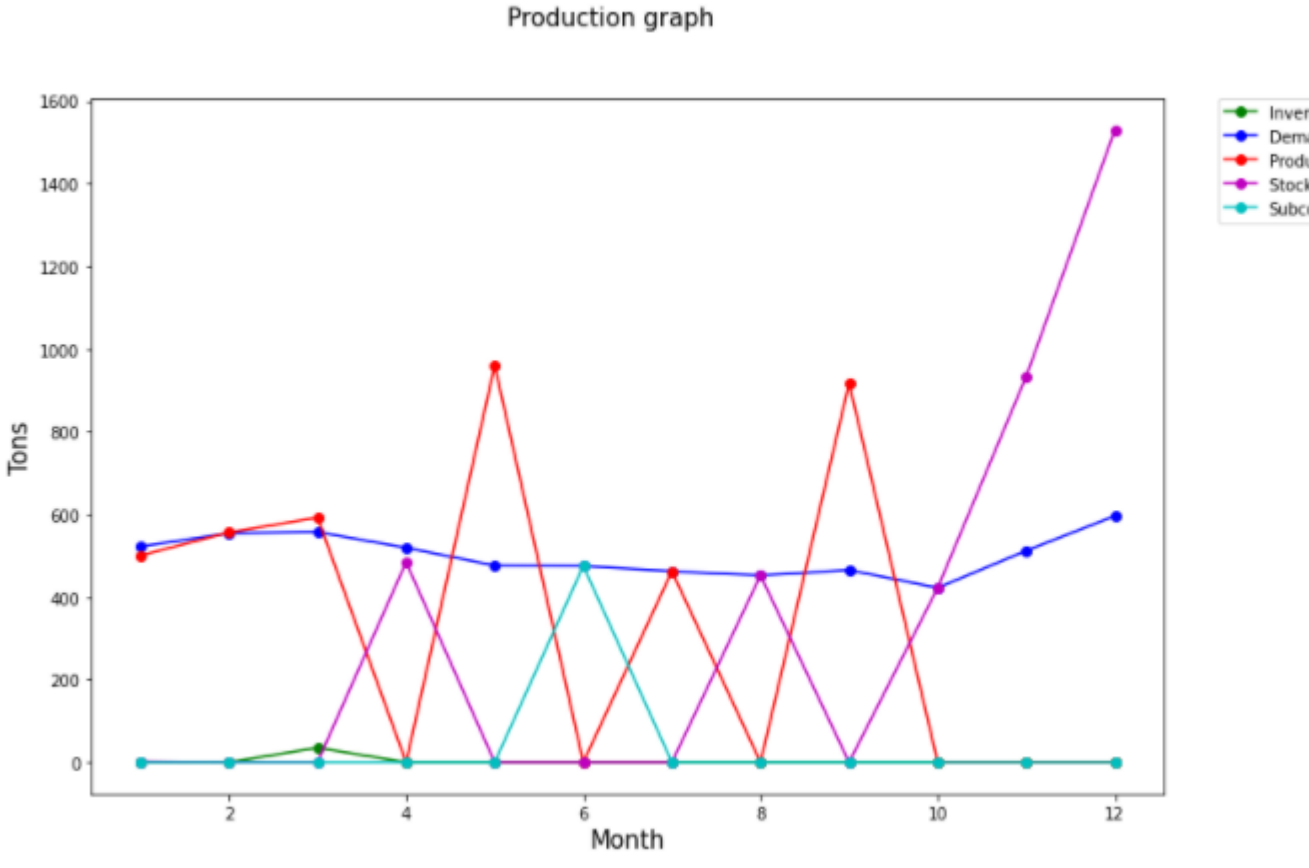
Objective function value: 38687.06

b. (25 points) Having solved your optimization model, for each following bullet point, make a single plot with the respective information as a function of the month:

- Monthly demand, monthly coffee production, monthly coffee subcontracting, coffee backlogs, and coffee inventory at the end of each month. What analysis can you make of this graph?

```
PSol={i:P[i].x for i in P.keys()}
ISol = {i:I[i].x for i in I.keys()}
SSol = {i:S[i].x for i in S.keys()}
CSol = {i:C[i].x for i in C.keys()}

def InvGraph(PSol,ISol,d,SSol,CSol):
    fig, ax = plt.subplots(figsize=(12,8))
    ax.plot(list(ISol.keys()),list(ISol.values()),'-o', color = 'g', label = "Inventory")
    ax.plot(list(d.keys()),list(d.values()),'-o', color = 'b', label = "Demand")
    ax.plot(list(PSol.keys()),list(PSol.values()),'-o', color = 'r', label = "Production")
    ax.plot(list(SSol.keys()),list(SSol.values()),'-o', color = 'm', label = "Stocking out")
    ax.plot(list(CSol.keys()),list(CSol.values()),'-o', color = 'c', label = "Subcontracting")
    ax.set_xlabel("Month", size=15)
    ax.set_ylabel("Tons", size=15)
    ax.legend(bbox_to_anchor=(1.05, 1), loc='best', borderaxespad=0.)
    plt.suptitle("Production graph" , fontsize=15)
```

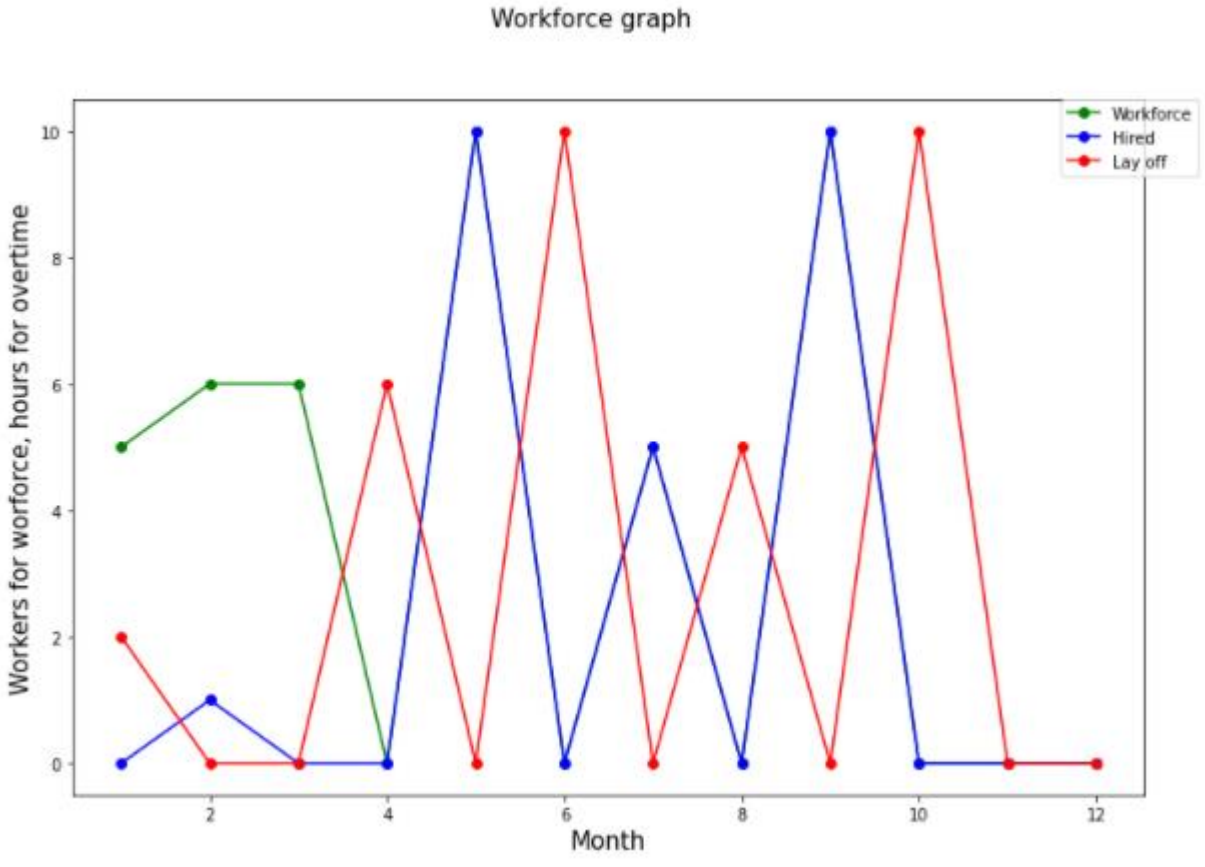
```
InvGraph(PSol,ISol,d,SSol,CSol)
```




- Monthly workforce, the monthly hirings, and the monthly layoffs.

```
WSol = {i:W[i].x for i in W.keys()}
HSol = {i:H[i].x for i in H.keys()}
FSol = {i:F[i].x for i in F.keys()}
OSol = {i:O[i].x for i in O.keys()}

def WforceGraph(WSol,HSol,FSol,OSol):
    fig, ax = plt.subplots(figsize=(12,8))
    ax.plot(list(WSol.keys()),list(WSol.values()),'-o', color = 'g', label = "Workforce")
    ax.plot(list(HSol.keys()),list(HSol.values()),'-o', color = 'b', label = "Hired")
    ax.plot(list(FSol.keys()),list(FSol.values()),'-o', color = 'r', label = "Lay off")
    ax.set_xlabel("Month", size=15)
    ax.set_ylabel("Workers for worforce, hours for overtime", size=15)
    ax.legend(bbox_to_anchor=(1.05, 1), loc='best', borderaxespad=0.)
    plt.suptitle("Workforce graph" , fontsize=15)
```

```
WforceGraph(WSol,HSol,FSol,OSol)
```




- Monthly number of extra hours worked.

```
def WforceHoursGraph(OSol):
    fig, ax = plt.subplots(figsize=(12,8))
    ax.plot(list(OSol.keys()),list(OSol.values()),'-o', color = 'm', label = "Overtime hours")
    ax.set_xlabel("Month", size=15)
    ax.set_ylabel("Hours for overtime", size=15)
    ax.legend(bbox_to_anchor=(1.05, 1), loc='best', borderaxespad=0.)
    plt.suptitle("Workforce hours graph" , fontsize=15)
```

```
WforceHoursGraph(OSol)
```


c. (10 points for graduate students/extra credit) From the original problem, assume that during the 12-month planning horizon the company can only produce up to 4 times (months) since it must close the production plant for maintenance purposes for the duration of the other 8 months (the maintenance does not have to be executed in consecutive months). How would this condition change the original solution? Show the same 3 graphs and interpret.

New variable:

$$y_t: 1 \text{ if produced in month } t \in T, 0 \text{ otherwise.}$$

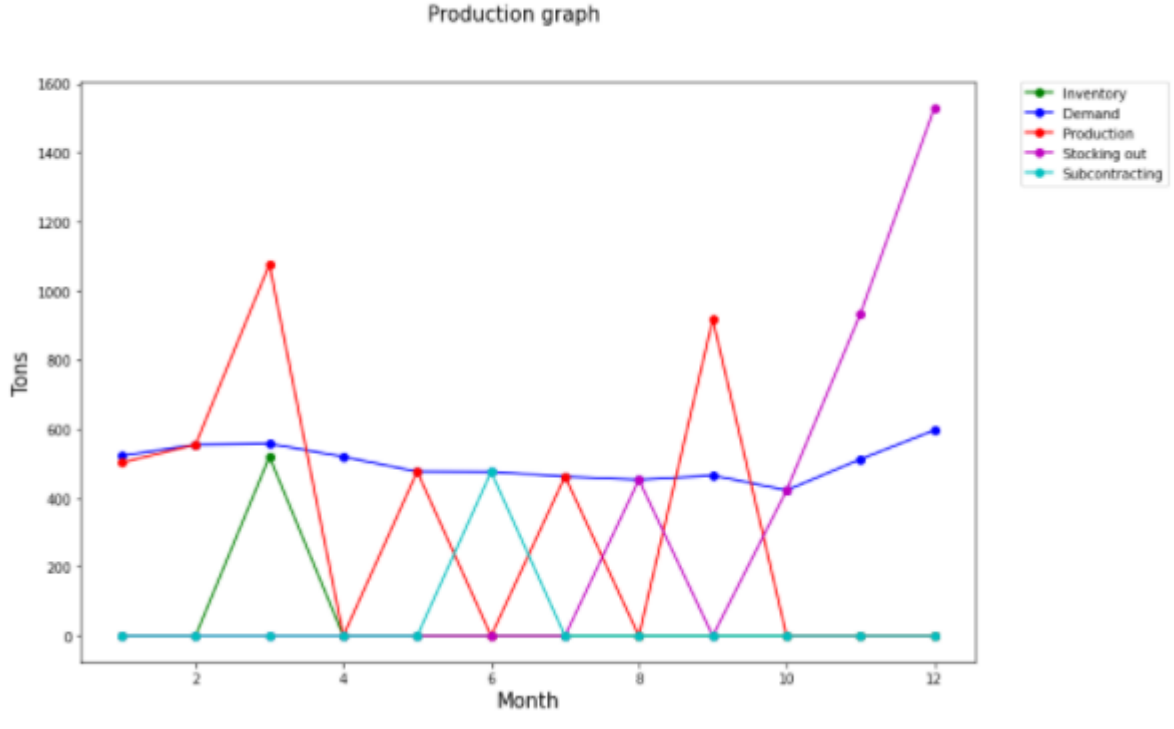
Necessary constraints:

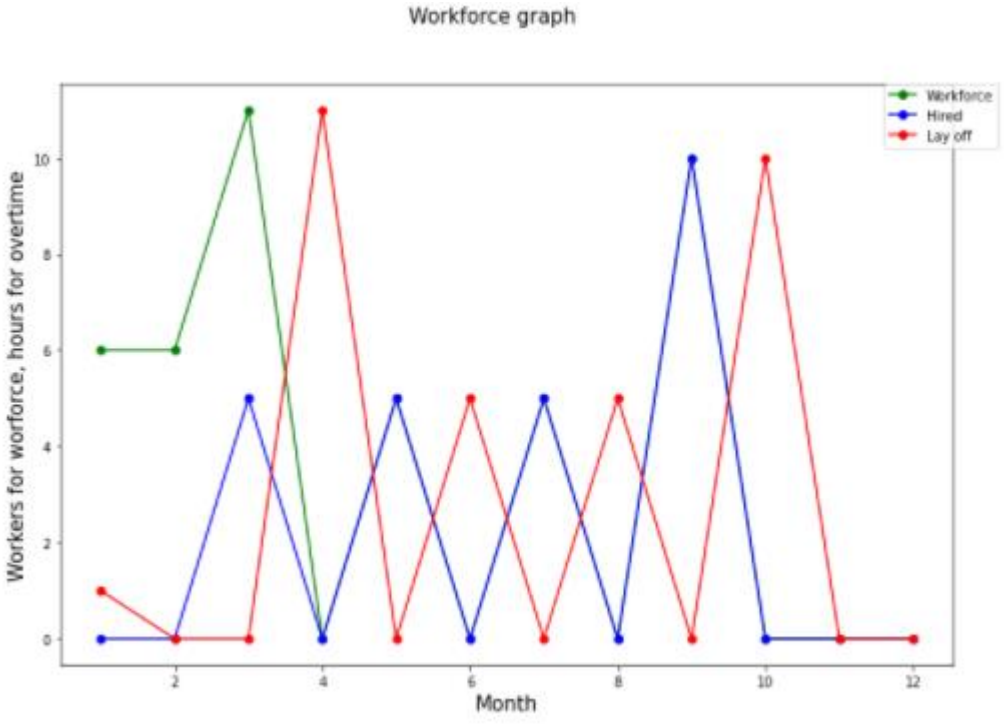
$$\text{bigNumber} \cdot y_t \geq P_t$$
$$y_t - 1 \leq P_t$$
$$\sum_{t \in T} y_t \leq 4$$
$$y_t \in \{0,1\} \; \forall \; t \in T$$

```
y = {t:my_model.addVar(vtype=GRB.BINARY,name="y_"+str(t)) for t in M}
my_model.update()

for t in M:
    my_model.addConstr(my_model.getVarByName('P_'+str(t)) <= my_model.getVarByName('y_'+str(t))*1e8)
    my_model.update()
    my_model.addConstr(my_model.getVarByName('P_'+str(t)) >= my_model.getVarByName('y_'+str(t)))
    my_model.update()

my_model.addConstr(quicksum(my_model.getVarByName('y_'+str(t)) for t in M)<=4)

my_model.update()

my_model.optimize()
```

Note: Here we added the constraints after we solved the model on a different Jupyter cell. That's why we call the variable by its name while creating the new constraints.

Objective function value: 38701.520000000004







d. (10 points extra credit everyone) From the original problem, assume that company can produce coffee once for a 3 consecutive month window (e.g., if the company produces coffee in January, it can't produce again until April). How would this condition change the original solution? Show the same 3 graphs and interpret.

Without considering part c.

New variable:

$$y_t: 1 \text{ if produced in month } t \in T, 0 \text{ otherwise.}$$

Necessary constraints:

$$\text{bigNumber} \cdot y_t \geq P_t$$

$$y_t - 1 \leq P_t$$

$$\sum_{s=t}^{t+2} y_s \leq 1 \; \forall \; t \in T \mid t \leq |T| - 2$$

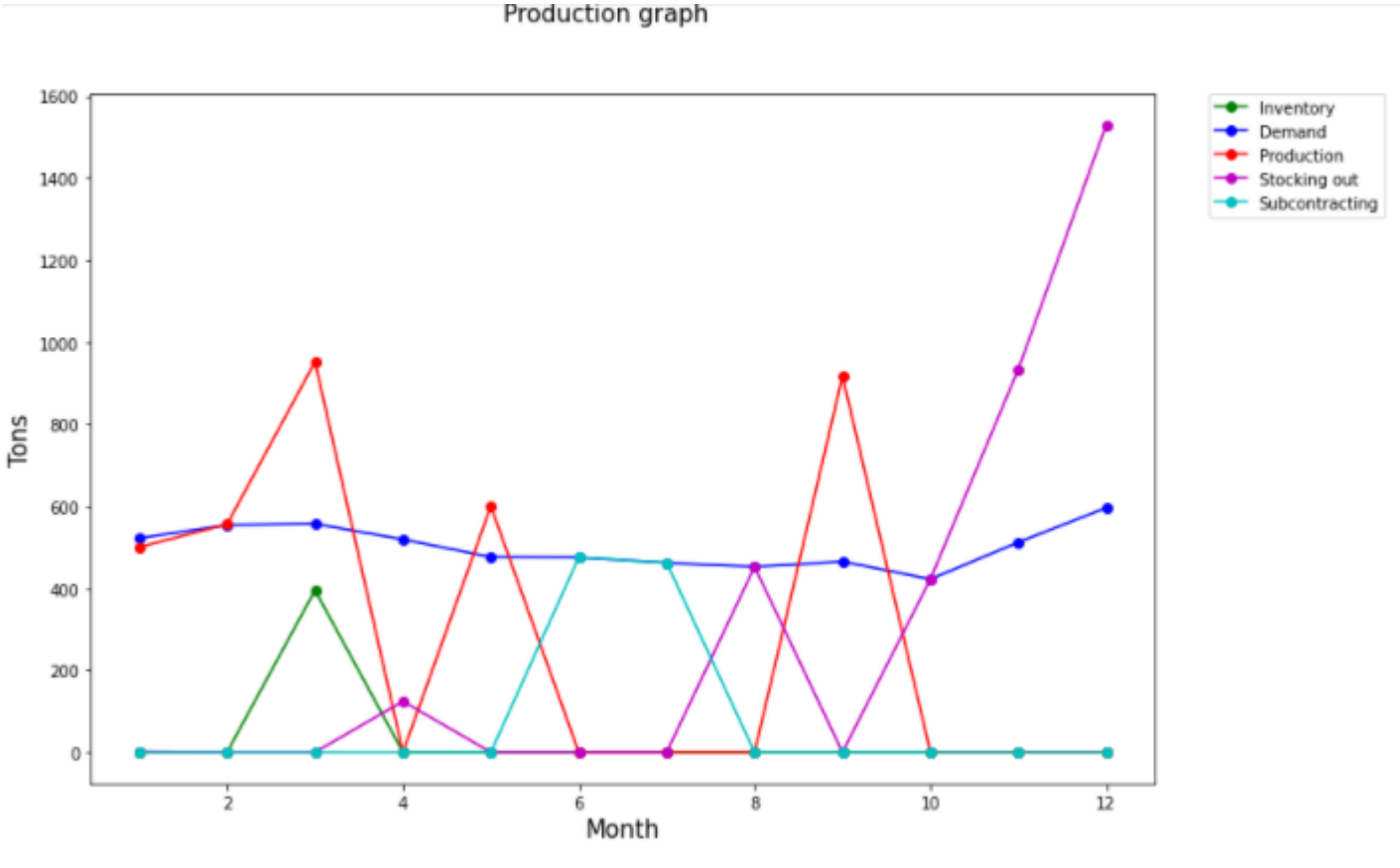
$$y_t \in \{0,1\} \; \forall \; t \in T$$

```
#If produced in a month
y = {t:my_model.addVar(vtype=GRB.BINARY,name="y_"+str(t)) for t in M}
```

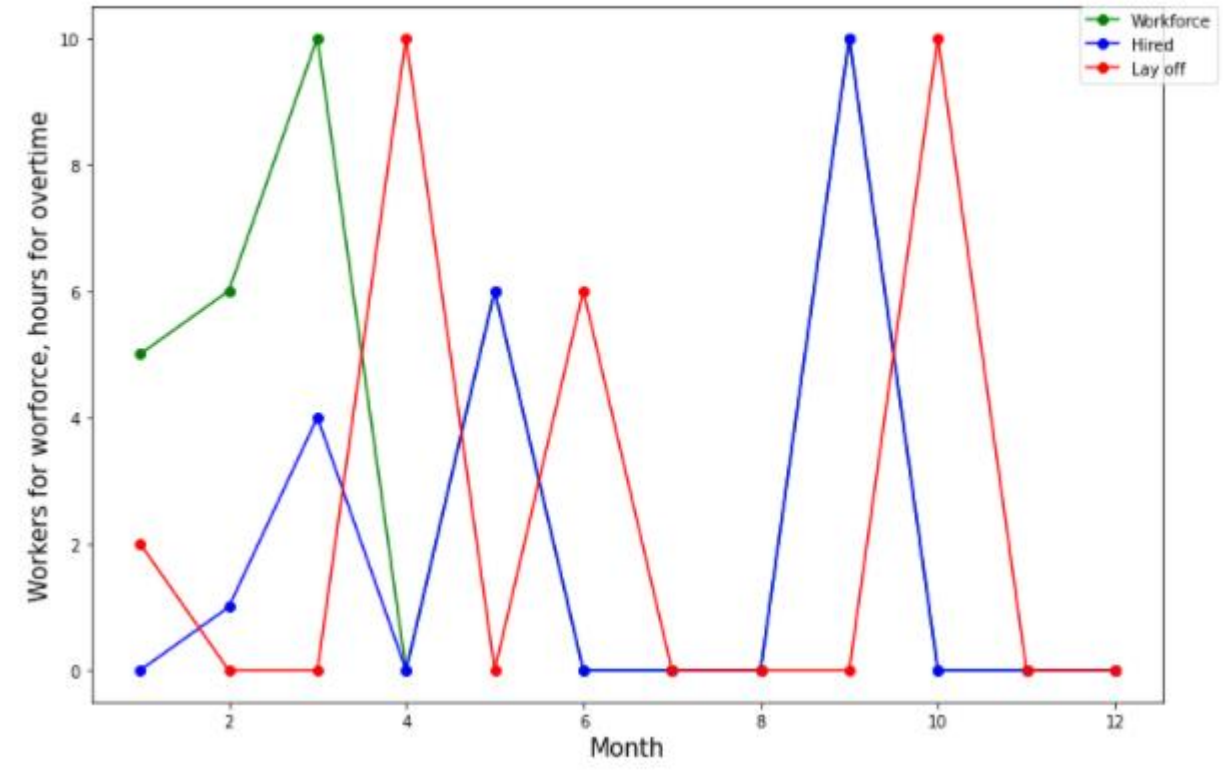
```
#Production binary relation
for t in M:
    my_model.addConstr(P[t] <= y[t]*1e8)
    my_model.addConstr(P[t] >= y[t])

#Not producing in 3 consecutive months
for t in range(1,M[-3]):
    my_model.addConstr(quicksum(y[s] for s in range(t,t+3)) <= 1)
```

Note: Here we copied the original model from a. so the new constraints could be added as usual.

Objective function value: 39039.56



Workforce graph




Workforce hours graph