Exercises - Section 4: Lecture 14 – Building a Model - Solution

1. A real estate investment company has an investment budget that they want to spend in a bunch of cities, in a way that maximizes the predicted increase in value. To diversify, they want to limit their spending in each region, and they also have set minimum and maximum spending limits in each city.

   A dataset for this question is given in file `question_one_data.json`. The mathematical and gurobipy models are described in the lessons. Create a file with the gurobipy model, and solve it.

   Solution:

   The model for this problem is given in the lessons, and the code (repeated here) is available in file `4b.4 question 1.py`:

```
import gurobipy as gp
from gurobipy import GRB

import json

with open("question_one_data.json", "r") as f:
    data = json.load(f)

B = data["B"]
N = data["N"]
P = data["profits"]
L = data["L"]
R = data["R"]
Mmax = data["M"]
Mmin = data["m"]
cities = data["cities"]

m=gp.Model("investment")

# Variables: Money (in millions of dollars) invested in each
city
x = m.addVars(N, lb=Mmin, ub=Mmax, name="x")


# Objective: Maximize predicted increase in value
m.setObjective(gp.quicksum(P[j]*x[j] for j in range(N)),
gp.GRB.MAXIMIZE)
# alternative code: m.setObjective(x.prod(P), GRB.MAXIMIZE)
```

```
# Constraint: Spend no more than B
m.addConstr(gp.quicksum(x[j] for j in range(N)) <= B,
name="budget")
# alternative code: m.addConstr(x.sum() <= B)

# Constraint: Limit spending in each region
m.addConstrs((gp.quicksum(x[j] for j in R[r]) <= L[r] for r in
range(len(R))), name="region")

# Constraint: Limit spending in each city
# already accounted for by the lb and ub parameters in the
variable declaration

# Constraint: Non-negativity
# already accounted for in the variable declaration


m.optimize()

print("***************** Solution *****************")
print(f"Total predicted value increase: {round(m.ObjVal, 2)}")
for i in range(N):
    print(f"Invest {x[i].X} million dollars in city
{cities[i]}.")
```

The optimal solution is to invest \$8 million in Atlanta, \$2 million in Boston, \$4 million in Charlotte, and \$6 million in Detroit, with a predicted value increase of \$1.26 million.

2. A metal alloy manufacturer has two processes that it can use to produce alloy. Operating Process 1 costs \$28 per ton of alloy created, and operating Process 2 costs \$53 per ton of alloy created. The plant would like to produce its yearly requirement of 3400 tons of alloy at minimum cost, subject to the requirement that at most two thirds of its production can be from Process 1. However, the law limits the amount of pollution the plant can put out to at most 200,000 pounds per year. If the plant goes beyond that limit, it will be shut down by the government. Operating Process 1 produces 90 pounds of pollution per ton of alloy created, and operating Process 2 produces 50 pounds of pollution per ton of alloy created.

Create mathematical and gurobipy models that the manufacturer can use to find the minimum-cost way of producing its annual requirement of alloy without going over the pollution limit. Solve the gurobipy model.

Solution:

ENGLISH	**MATH**	**GUROBIPY** `m = gp.Model("metal")`
<u>Variables</u> Tons of alloy produced by each process	x_1, x_2	`x = m.addVars(range(1,3), lb=0, vtype=gp.GRB.CONTINUOUS, name="x")`
<u>Objective</u> Minimize production cost	Minimize $28x_1 + 53x_2$	`m.setObjective(28*x[1] + 53*x[2], gp.GRB.MINIMIZE)`
<u>Constraints</u> Produce yearly requirement	$x_1 + x_2 \geq 3400$	`m.addConstr(x[1] + x[2] >= 3400)`

Process 1 is at most two thirds the total production	$x_1 \leq \frac{2}{3}(x_1 + x_2)$
	or
	$x_1 \leq 2x_2$
		`m.addConstr(x[1] <= 2*(x[1]+x[2])/3)`
		or
		`m.addConstr(x[1] <= 2*x[2])`

Pollution limit	$90x_1 + 50x_2 \leq 200000$	`m.addConstr(90*x[1] + 50*x[2] <= 200000)`
Non-negativity	$x_1, x_2 \geq 0$	`# part of variable definition`

The optimal solution is to produce 750 tons using Process 1 and 2650 tons using Process 2, at a total cost of \$161,450.

See `4b.4 question 2.py` for the gurobipy code.

3. A large new furniture store wants to decide how much of its floor space will be devoted to beds, bookcases, chairs (dining room, living room, and office), coffee tables, couches, desks, dining tables, and dressers.

In order to not be known as a specialized store, it needs to devote at least 20% of its floor space to bedroom furniture, at least 20% to living room furniture, at least 15% to dining room furniture, and at least 10% to office furniture.

At least 10% of the floor space must be devoted to beds, at least 50% to seating (chairs, couches), at least 10% to tables (coffee, dining), and at least 5% to storage and work (bookcases, desks, dressers).

Because dining room tables and chairs are often purchased together (and tables take 1.5 times the space of chairs on display), the amount of floor space devoted to dining room tables must be 1.5 times the floor space devoted to dining room chairs.

Every type of furniture must get at least 5% of the floor space.

The table below shows how each type of furniture is categorized, and the marginal profit of increasing the size of space devoted to each type of furniture.

Index	Furniture type	Categories	Marginal annual profit per extra percent of floor space (scaled)
1	Beds	Bedroom	10
2	Bookcases	Office, Storage/Work	7
3	Chairs, dining room	Dining Room, Seating	12
4	Chairs, living room	Living Room, Seating	9
5	Chairs, office	Office, Seating	3
6	Coffee tables	Living Room, Tables	5
7	Couches	Living Room, Seating	8
8	Desks	Office, Storage/Work	6
9	Dining tables	Dining Room, Tables	13
10	Dressers	Bedroom, Storage/Work	4

Create mathematical and gurobipy models that the manufacturer can use to find the most profitable way to divide its floor space. Solve the gurobipy model.

Solution 1: Hard-coded data

ENGLISH	**MATH**	**GUROBIPY** `m = gp.Model("furniture")`
Data Number of furniture items	N	`N = 10`
Variables Fraction of floor space for each furniture type	x_i	`x = m.addVars(range(1,N+1), lb=0.05,ub=1, vtype=gp.GRB.CONTINUOUS, name="x")`
Objective Maximize marginal annual profit	Maximize $\sum_{i=1}^{10} p_i x_i$	`m.setObjective(10*x[1] + 7*x[2] + 12*x[3] + 9*x[4] + 3*x[5] + 5*x[6] + 8*x[7] + 6*x[8] + 13*x[9] + 4*x[10], gp.GRB.MAXIMIZE)`
Constraints Floor space minimums: ...Bedroom furniture	$x_1 + x_{10} \geq 0.2$	`m.addConstr(x[1] + x[10] >= 0.2)`
...Living Room furniture	$x_4 + x_6 + x_7 \geq 0.2$	`m.addConstr(x[4] + x[6] + x[7] >= 0.2)`
...Dining Room furniture	$x_3 + x_9 \geq 0.15$	`m.addConstr(x[3] + x[9] >= .15)`
...Office furniture	$x_2 + x_5 + x_8 \geq 0.1$	`m.addConstr(x[2] + x[5] + x[8] >= .1)`
...Beds	$x_1 \geq 0.1$	`m.addConstr(x[1] >= .1)`
...Seating	$x_3 + x_4 + x_5 + x_7 \geq 0.5$	`m.addConstr(x[3] + x[4] + x[5] + x[7] >= .5)`
...Tables	$x_6 + x_9 \geq 0.1$	`m.addConstr(x[6] + x[9] >= .1)`
...Storage/Work furniture	$x_2 + x_8 + x_{10} \geq 0.05$	`m.addConstr(x[2] + x[8] + x[10] >= .05)`
Dining tables-to-dining chairs ratio	$x_9 = 1.5x_3$	`m.addConstr(x[9] == 1.5*x[3])`
Entire floor space must be used	$\sum_{i=1}^{10} x_i = 1$	`m.addConstr(sum(x[i] for i in range(1,N+1)) == 1)`
At least 5% for each type of furniture	$x_i \geq 0.05$ for all i	`# this is ensured by setting lb=0.05 in the variables definition`
Non-negativity	Implied by previous constraint	`# Already implied by each variable having a lb=0.05`

See `4b.4 question 3.py` for the gurobipy code.

Solution 2: General model with data read from file

ENGLISH	MATH	GUROBIPY	`m = gp.Model("furniture")`
<u>Data</u>		`# Read from file`	
List of furniture types	F	`Furniture = data["Furniture"]`	
Minimum floor space for each furniture type	m_i	`F_mins = data["Fmins"]`	
Marginal annual profit of each furniture type per extra percent of floor space	p_i	`M_Profit = data["MarginalProfit"]`	
List of furniture categories	C	`Categories = data["Categories"]`	
Minimum floor space for each furniture category	l_j	`C_mins = data["Cmins"]`	
List of furniture categories for each furniture type	L_i	`Classifications = data["Classifications"]`	
Extra constraint data (dining tables & chairs)	n/a (extra constraint will be hard-coded)	`Extra = data["ExtraGeqConstraints"]`	
<u>Variables</u> Fraction of floor space for each furniture type	x_i	`x = m.addVars(Furniture, lb=F_mins, ub=1, vtype=gp.GRB.CONTINUOUS, name="x")`	
<u>Objective</u> Maximize marginal annual profit	Maximize $\sum_{i \in F} p_i x_i$	`m.setObjective(x.prod(M_Profit), gp.GRB.MAXIMIZE)`	
<u>Constraints</u> Minimum floor space for each category	$\sum_{i: j \in L_i} x_i \geq l_j$ for all categories j	`m.addConstrs(sum(x[i] for i in Furniture if j in Classifications[i]) >= C_mins[j] for j in Categories)`	
Entire floor space must be used	$\sum_{i \in F} x_i = 1$	`m.addConstr(x.sum() == 1)`	
At least 5% for each type of furniture, at least 10% beds	$x_i \geq m_i$ for all furniture types i	`# this is ensured by setting lb=F_mins in the definition of the variables`	
Dining tables-to-dining chairs ratio	$x_{\text{dining tables}} \geq 1.5 x_{\text{dining room chairs}}$	`# This additional >= constraint doesn't fit nicely into any of the others, so its information is in the list "ExtraGeqConstraints"`	

		`m.addConstrs(Extra[j][0] * x[Extra[j][1]] >= Extra[j][2] * x[Extra[j][3]] for j in Extra)`
Non-negativity	Not needed, because $x_i \geq m_i$ implies non-negativity	`# Already implied by each variable having a lb=0.05`

See `4b.4 question 3-v2.py` for the gurobipy code.

Whichever model you used, the optimal solution is:

Optimal allocation of floor space for the ten furniture types:

15% Beds
5% Bookcases
10% Chairs (dining room)
30% Chairs (living room)
5% Chairs (office)
5% Coffee tables
5% Couches
5% Desks
15% Dining tables
5% Dressers

The optimal marginal profit is 9 (note that this is scaled; it's not just $9!).

4. A regional recycling center collects used paper, glass, and plastic. Every month, it sells the paper, glass, and plastic to large recycling companies (and those companies recycle the materials). Each large recycling company purchases different combinations of materials. The table below shows the weight ratio that each company purchases materials to recycle, and the per-pound cost they pay (for example, Company A purchases paper, glass, and plastic in a ratio of 5 pounds of paper to 3 pounds of glass to 1 pound of plastic, and pays $2 for each 9 pounds of total materials).

Recycling Company	Pounds of paper per bundle	Pounds of glass per bundle	Pounds of plastic per bundle	Purchase price for bundle of paper, glass, and plastic
A	5 lbs	3 lbs	1 lb	$2 per 9 lbs bundle
B	7 lbs	3 lbs	2 lbs	$3 per 10 lbs bundle
C	2 lbs	1 lb	1 lb	$1 per 4 lbs bundle
D	3 lbs	1 lbs	4 lbs	$2 per 9 lbs bundle

In addition to bundles, the recycling center can also sell paper, glass, and plastic to small recyclers for $0.05 per pound of paper, $0.03 per pound of glass, and $0.02 per pound of plastic.

To maintain their business relationship, the recycling center must sell at least 100 full bundles to each large recycler. Beyond that minimum, either full or partial bundles may be sold.

(a) This month, the recycling center has collected 15,000 pounds of paper, 7,000 pounds of glass, and 14,000 pounds of plastic. Create mathematical and gurobipy models that the recycling center can use to maximize its revenue for selling all that it collected this month. Solve the gurobipy model.

Solution 1: Hard-coded data

<table>
<tr><th>ENGLISH</th><th>MATH</th><th>GUROBIPY <code>m = gp.Model("recycle")</code></th></tr>
<tr><td><u>Data</u>
List of materials
List of companies</td><td>M
C</td><td><code>M = ["Paper", "Glass", "Plastic"]</code>
<code>C = ["A", "B", "C", "D"]</code></td></tr>
<tr><td><u>Variables</u>
Bundles sold to each large company

Pounds of each material sold to small companies</td><td>x_i

y_j</td><td><code>x = m.addVars(C, lb=100, vtype=gp.GRB.CONTINUOUS, name="x")</code>

<code>y = m.addVars(M, vtype=gp.GRB.CONTINUOUS, name="y")</code></td></tr>
</table>

<u>Objective</u> Maximize revenue	Maximize $2x_A + 3x_B + 1x_C + 2x_D + 0.05y_{\text{Paper}} + 0.03y_{\text{Glass}} + 0.02y_{\text{Plastic}}$	`m.setObjective(2*x["A"] + 3*x["B"] + 1*x["C"] + 2*x["D"] + 0.05*y["Paper"] + 0.03*y["Glass"] + 0.02*y["Plastic"], GRB.MAXIMIZE)`
<u>Constraints</u> Sell all collected paper (in bundles and to small companies)	$5x_A + 7x_B + 2x_C + 3x_D + y_{\text{Paper}} = 15000$	`m.addConstr(5*x["A"] + 7*x["B"] + 2*x["C"] + 3*x["D"] + y["Paper"] == 15000)`
Sell all collected glass (in bundles and to small companies)	$3x_A + 3x_B + 1x_C + 1x_D + y_{\text{Glass}} = 7000$	`m.addConstr(3*x["A"] + 3*x["B"] + 1*x["C"] + 1*x["D"] + y["Glass"] == 7000)`
Sell all collected plastic (in bundles and to small companies)	$1x_A + 2x_B + 1x_C + 4x_D + y_{\text{Plastic}} = 14000$	`m.addConstr(1*x["A"] + 2*x["B"] + 1*x["C"] + 4*x["D"] + y["Plastic"] == 14000)`
At least 100 bundles must be sold to each large company	$x_A \geq 100$ $x_B \geq 100$ $x_C \geq 100$ $x_D \geq 100$	`# This is ensured by setting lb=100 in the definition of the x-variables`
Non-negativity	all $y_j \geq 0$ Not needed for x, because $x_i \geq 100$ implies non-negativity	`# Implied in variable declarations`

See `4b.4 question 4a.py` for the gurobipy code.

Solution 2: General model with data read from file

ENGLISH	**MATH**	**GUROBIPY** `m = gp.Model("recycle")`

<u>Data</u>
List of materials
List of companies
Amount of each material in each bundle
Minimum bundles sold to each company
Bundle price from each company
Selling price to small recyclers of each material
Amount of each material collected
	
	M
	C
	a_{ij}
	m_i
	b_j
	s_j
	r_j
		`# Read from file`
		`Materials = data["Materials"]`
		`Companies = data["Companies"]`
		`Bundles = data["Bundles"]`
		
		`Minimums = data["Minimums"]`
		
		`BundlePrices = data["BundlePrices"]`
		`SmallPrices = data["SmallPrices"]`
		
		`Collected = data["Collected"]`

<u>Variables</u>
Bundles sold to each large company
 	
Pounds of each material sold to small companies
	
	x_i
	
	y_j
		
		`x = m.addVars(Companies, lb=Minimums, vtype=gp.GRB.CONTINUOUS, name="x")`
		
		`y = m.addVars(Materials, vtype=gp.GRB.CONTINUOUS, name="y")`

<u>Objective</u> Maximize revenue	 Maximize $\sum_{i\in C} b_i x_i + \sum_{j\in M} s_j y_j$	 `m.setObjective( x.prod(BundlePrices) + y.prod(SmallPrices), gp.GRB.MAXIMIZE)`

<u>Constraints</u>
Sell all collected materials (in bundles and to small companies)
 	
At least 100 bundles must be sold to each large company
 	
Non-negativity
	
	$\sum_{i\in C} a_{ij} x_i + y_j = r_j$ for all materials j
	
	$x_i \geq m_i$ for all large companies i
	
	all $y_j \geq 0$ for all materials j
	
	Not needed for x, because $x_i \geq m_i$
		
		`m.addConstrs(sum(Bundles[i][j] * x[i] for i in Companies) + y[j] == Collected[j] for j in Materials))`
		
		`# This is ensured by setting lb=Minimums in the definition of the x-variables`
		
		`# Implied in variable declarations`

	implies non-negativity	

See `4b.4 question 4-v2.py` for the gurobipy code.

The optimal revenue is $8,785.80:

Sell 100 bundles to company A.

Sell 100 bundles to company B.

Sell 2820 bundles to company C.

Sell 2720 bundles to company D.

Sell 860 pounds of glass to small recyclers.

(b) Next month, the recycling center collects 18,000 pounds of paper, 5,000 pounds of glass, and 10,000 pounds of plastic. Solve your gurobipy model from part (a) with this new data, to determine how the recycling center can maximize its revenue this month.

Solution 1: Hard-coded data

Make the following changes in the model:

ENGLISH	**MATH**	**GUROBIPY**
<u>Constraints</u> Sell all collected paper (in bundles and to small companies)	$5x_A + 7x_B + 2x_C + 3x_D + y_{\text{Paper}} = 18000$	`m.addConstr(5*x["A"] + 7*x["B"] + 2*x["C"] + 3*x["D"] + y["Paper"] == 18000)`
Sell all collected glass (in bundles and to small companies)	$3x_A + 3x_B + 1x_C + 1x_D + y_{\text{Glass}} = 5000$	`m.addConstr(3*x["A"] + 3*x["B"] + 1*x["C"] + 1*x["D"] + y["Glass"] == 5000)`
Sell all collected plastic (in bundles and to small companies)	$1x_A + 2x_B + 1x_C + 4x_D + y_{\text{Plastic}} = 10000$	`m.addConstr(1*x["A"] + 2*x["B"] + 1*x["C"] + 4*x["D"] + y["Plastic"] == 10000)`

See `4b.4 question 4b.py` for the gurobipy code.

Solution 2: General model with data read from file

No changes needed except in data file (and in specifying the correct data file in gurobipy).

See `4b.4 question 4-v2.py` for the gurobipy code.

The optimal revenue is reduced to $7,181.00:

Sell 100 bundles to company A.

Sell 860 bundles to company B.

Sell 100 bundles to company C.

Sell 2020 bundles to company D.

Sell 5220 pounds of paper to small recyclers.

(c) Suppose the recycling center was able store two months of recyclables, so it could sell both months' recyclables at once. Solve your gurobipy model from part (a) with the combined two-month data, to determine how much additional revenue the recycling center could get if it had two months of storage capacity. (Note that the need to sell 100 bundles/month to each large recycler means that the recycling center needs to sell at least 200 bundles to each large recycler over this two-month time period.)

Solution 1: Hard-coded data

Make the following changes in the model:

ENGLISH	MATH	GUROBIPY
Variables Bundles sold to each large company	x_i	`x = m.addVars(C, lb=200, vtype=gp.GRB.CONTINUOUS, name="x")`
Constraints Sell all collected paper (in bundles and to small companies)	$5x_A + 7x_B + 2x_C + 3x_D + y_{\text{Paper}} = 33000$	`m.addConstr(5*x["A"] + 7*x["B"] + 2*x["C"] + 3*x["D"] + y["Paper"] == 33000)`
Sell all collected glass (in bundles and to small companies)	$3x_A + 3x_B + 1x_C + 1x_D + y_{\text{Glass}} = 12000$	`m.addConstr(3*x["A"] + 3*x["B"] + 1*x["C"] + 1*x["D"] + y["Glass"] == 12000)`
Sell all collected plastic (in bundles and to small companies)	$1x_A + 2x_B + 1x_C + 4x_D + y_{\text{Plastic}} = 24000$	`m.addConstr(1*x["A"] + 2*x["B"] + 1*x["C"] + 4*x["D"] + y["Plastic"] == 24000)`
At least 200 bundles must be sold to each large company	$x_A \geq 200$ $x_B \geq 200$ $x_C \geq 200$ $x_D \geq 200$	`# This is ensured by setting lb=200 in the definition of the x-variables`

See `4b.4 question 4c.py` for the gurobipy code.

Solution 2: General model with data read from file

No changes needed except in data file (and in specifying the correct data file in gurobipy).

See `4b.4 question 4-v2.py` for the gurobipy code.

The optimal revenue is from two months is $16,752.00:

Sell 200 bundles to company A.

Sell 2120 bundles to company B.

Sell 200 bundles to company C.

Sell 4840 bundles to company D.

Sell 2240 pounds of paper to small recyclers.

The optimal two-month revenue is $785.20 more than the sum of the two one-month solutions, so having the extra month of storage capacity would yield slightly less than $800 benefit over the two-month period.

5. The environmental regulators of a certain country (with very strong central government control) would like to make sure that the air over a city does not get too polluted, while at the same time making sure that businesses are not forced into bankruptcy by environmental regulations.

   To control the air pollution, the regulators have identified *C* companies to regulate. Each company *j* currently emits p_j tons of pollutants each year. To reduce the amount of pollution, they estimate that it will cost them d_j dollars per year per ton decrease. Each company *j* currently has operating expenses of c_j dollars per year; in order to maintain a reasonable level of profit, they must keep their expenses below m_j per year.

   Create mathematical and gurobipy models that the regulators can use to assign allowable pollution levels to each company that minimize the total expense of the companies while getting the total pollution to be no greater that *T* tons per year and while allowing each company to maintain its reasonable level of profit.

There are different solutions you might have come up with, depending on how variables are defined.

Solution 1: Variables for the amount of the pollution decrease for each company

ENGLISH	**MATH**	**GUROBIPY**	`m = gp.Model("pollution")`
<u>Data</u>		`# Read from file`	
Number of companies	C	`# C is a number`	
Yearly pollutant emission by each company	p_j	`# p is a list`	
	d_j	`# d is a list`	

Cost per ton to decrease each company's pollution
Yearly operating expense for each company
Maximum expenses for each company to maintain profit
Total pollution limit
	c_j
	
	m_j
	
	T
		`# c is a list`
		
		`# m is a list`
		
		`# T is a number`

<u>Variables</u> Pollution decrease for each company	x_j	`x = m.addVars(C, lb=p, vtype=gp.GRB.CONTINUOUS, name="x")`
<u>Objective</u> Minimize total expense	Minimize $\sum_{j=1}^{C} d_j x_j$	`m.setObjective(x.prod(d) gp.GRB.MINIMIZE)`

Constraints Keep each company's expenses low enough	$c_j + d_j x_j \leq m_j$ for all companies j	`m.addConstrs(c[j] + d[j]*x[j] <= m[j] for j in range(C)))`
Total pollution cannot be above limit	$\sum_{j=1}^{C}(p_j - x_j) \leq T$	`m.addConstr(sum(p[j]-x[j] for j in range(C)) <= T)`
Can't decrease more than current pollution	$x_j \leq p_j$ for all companies j	`# specified by ub=p in variable declarations`
Non-negativity	$x_j \geq 0$ for all companies j	`# Implied in variable declarations`

Solution 2: Variables for the amount of pollution allowed to each company

ENGLISH	MATH	GUROBIPY `m = gp.Model("pollution")`
Data		`# Read from file`
Number of companies	C	`# C is a number`
Yearly pollutant emission by each company	p_j	`# p is a list`
Cost per ton to decrease each company's pollution	d_j	`# d is a list`
Yearly operating expense for each company	c_j	`# c is a list`
Maximum expenses for each company to maintain profit	m_j	`# m is a list`
Total pollution limit	T	`# T is a number`
Variables Pollution allowed to each company	y_j	`y = m.addVars(C, ub=p, vtype=gp.GRB.CONTINUOUS, name="y")`
Objective Minimize total expense	Minimize $\sum_{j=1}^{C} d_j(p_j - y_j)$	`m.setObjective(sum(d[j] * (p[j]-y[j]) for j in range(C)), gp.GRB.MINIMIZE)`
Constraints Keep each company's expenses low enough	$c_j + d_j(p_j - y_j) \leq m_j$ for all companies j	`m.addConstrs(c[j] + d[j]*(p[j]-y[j]) <= m[j] for j in range(C)))`
Total pollution cannot be above limit	$\sum_{j=1}^{C} y_j \leq T$	`m.addConstr(y.sum() <= T)`
Can't pollute more than current level	$y_j \leq p_j$ for all companies j	`# specified by ub=p in variable declarations`
Non-negativity	$y_j \geq 0$ for all companies j	`# Implied in variable declarations`

NOTES: