Optimization I

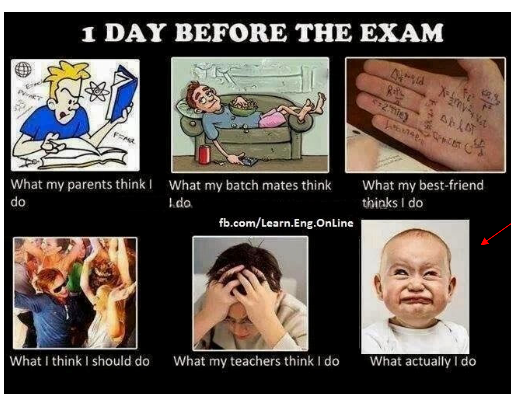
Janne Kettunen

1

1

Final Exam Date

- Final exam is scheduled for October 21, 4:30 pm – 7:00 pm in Duques 258 (at your normal class time and classroom)



Should not be the case for you!

2

2

Review of LP Workshop

3

3

Key Takeaways

- Decisions should not be made based on intuition or gut feeling
- Decision models do not also provide the ultimate solution, you still need to make your managerial judgement in light of the information they provide
- Sensitivity analysis and shadow prices are useful in pricing additional resources
- Optimization can be conducted over multiple scenarios to find the optimal resource allocation strategy on average

4

4

Solving RBC with Python and Gurobi

5

5

RBC Optimization Model

- Maximize profit contribution
  - Subject to
    - Demand constraints
    - Supply constraints
    - Quality constraints
    - Non-negativity constraints

- Maximize 246.67A_w+198A_j+222A_p + 246.67B_w+198B_j+222B_p
  Subject to

$$A_w + B_w \leq 14{,}400$$
$$A_j + B_j \leq 1{,}000$$
$$A_p + B_p \leq 2{,}000$$
$$A_w + A_j + A_p \leq 600$$
$$B_w + B_j + B_p \leq 2{,}400$$
$$9A_w + 5B_W \geq 8(A_w + B_w)$$
$$9A_j + 5B_j \geq 6(A_j + B_j)$$
$$A_w, B_w, A_j, B_j, A_p, B_p \geq 0$$

6

6

RBC Optimization, Python and Gurobi

Max $246.67A_w + 198A_j + 222A_p + 246.67B_w + 198B_j + 222B_p$

Subject to

$$A_w + B_w \leq 14{,}400$$
$$A_j + B_j \leq 1{,}000$$
$$A_p + B_p \leq 2{,}000$$
$$A_w + A_j + A_p \leq 600$$
$$B_w + B_j + B_p \leq 2{,}400$$
$$9A_w + 5B_W \geq 8(A_w + B_w)$$
$$9A_j + 5B_j \geq 6(A_j + B_j)$$

$$A_w, B_w, A_j, B_j, A_p, B_p \geq 0$$

“RBC (plain).ipynb”

```python
import gurobipy as gp
from gurobipy import GRB

# Model
m = gp.Model("RBC")

# Create decision variables for tomatoes usage
aw = m.addVar(name="aw")
aj = m.addVar(name="aj")
ap = m.addVar(name="ap")
bw = m.addVar(name="bw")
bj = m.addVar(name="bj")
bp = m.addVar(name="bp")

# The objective is to maximize the profit contribution
obj = (4.44/18*1000)*aw+198*aj+222*ap+(4.44/18*1000)*bw+198*bj+222*bp
m.setObjective(obj, GRB.MAXIMIZE)

# Demand constraints
con1 = m.addConstr(aw+bw<=14400, name='w_dem')
con2 = m.addConstr(aj+bj<=1000, name='j_dem')
con3 = m.addConstr(ap+bp<=2000, name='p_dem')

# Supply constraints
con4 = m.addConstr(aw+aj+ap<=600, name='a_sup')
con5 = m.addConstr(bw+bj+bp<=2400, name='b_sup')

# Quality constraints
con6 = m.addConstr(9*aw+5*bw>=8*(aw+bw), name='w_qual')
con7 = m.addConstr(9*aj+5*bj>=6*(aj+bj), name='j_qual')

# Non-negativity constraints
con8 = m.addConstr(aw>=0)
con9 = m.addConstr(aj>=0)
con10 = m.addConstr(ap>=0)
con11 = m.addConstr(bw>=0)
con12 = m.addConstr(bj>=0)
con13 = m.addConstr(bp>=0)

# Solve
m.optimize()

# Print optimal value of the objective function
print('\nProfit Contribution: %g' % m.objVal)
# Print optimal values for the decision variables
print('\nDecision variables:')
for v in m.getVars():
    print('%s = %g' % (v.varName, v.x))
```

7

Sensitivity Analysis Using Python and Gurobi “RBC (sensitivity).ipynb”

```python
# Create table for decision variables' sensitivity analysis
decision_var = OrderedDict([
    ('Name', ['aw', 'aj', 'ap', 'bw', 'bj', 'bp']),
    ('Final Value', [aw.x, aj.x, ap.x, bw.x, bj.x, bp.x]),
    ('Reduced Cost', [aw.RC, aj.RC, ap.RC, bw.RC, bj.RC, bp.RC]),
    ('Obj Coeff', [(4.44/18*1000), 198, 222, (4.44/18*1000), 198, 222]),
    ('Upper Range', [aw.SAObjUp, aj.SAObjUp, ap.SAObjUp, bw.SAObjUp, bj.SAObjUp, bp.SAObjUp]),
    ('Lower Range', [aw.SAObjLow, aj.SAObjLow, ap.SAObjLow, bw.SAObjLow, bj.SAObjLow, bp.SAObjLow])
])


# Create table for constraints' sensitivity analysis
constraint = OrderedDict([
    ('Name', ['w_dem', 'j_dem', 'p_dem', 'a_sup', 'b_sup', 'w_qual', 'j_qual']),
    ('Shadow Price', [con1.Pi, con2.Pi, con3.Pi, con4.Pi, con5.Pi, con6.Pi, con7.Pi]),
    ('RHS Coeff', [14400, 1000, 2000, 600, 2400, 0, 0]),
    ('Slack', [con1.Slack, con2.Slack, con3.Slack, con4.Slack, con5.Slack, con6.Slack, con7.Slack]),
    ('Upper Range', [con1.SARHSUp, con2.SARHSUp, con3.SARHSUp, con4.SARHSUp, con5.SARHSUp, con6.SARHSUp, con7.SARHSUp]),
    ('Lower Range',
     [con1.SARHSLow, con2.SARHSLow, con3.SARHSLow, con4.SARHSLow, con5.SARHSLow, con6.SARHSLow, con7.SARHSLow])
])

# Print sensitivity analysis tables for decision variables and constraints
print('\n')
print(pd.DataFrame.from_dict(decision_var))
print('\n')
print(pd.DataFrame.from_dict(constraint))
```

8

8

Differences between Excel's and Gurobi's Sensitivity Analysis Results

Adjustable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
B4	Grade A Whole	525	0	246.6667	463.11	64.89
C4	Grade A Juice	75	0	198.0000	64.89	463.11
D4	Grade A Paste	0	$\boxed{0}$	222.0000	97.33	1E+30
B5	Grade B Whole	175	0	246.6667	1389.33	64.89
C5	Grade B Juice	225	0	198.0000	42.96	154.37
D5	Grade B Paste	2,000	0	222.0000	1E+30	48.33

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
E4	Grade A Total Required	600	**271.00**	600	**600.00**	466.67
E5	Grade B Total Required	2,400	173.67	2,400	466.67	200.00
B6	Total Production Whole	700	0.00	14,400	1E+30	13700.00
C6	Total Production Juice	300	0.00	1,000	1E+30	700.00
D6	Total Production Paste	2,000	48.33	2,000	200.00	466.67
B12	Total Quality Whole	5,600	-24.33	0	466.67	600.00
C12	Total Quality Juice	1,800	-24.33	0	1400.00	200.00
D12	Total Quality Paste	10,000	-24.33	0	1400.00	0.00

The reduced cost can be wrong in Excel!

You can check this by including a constraint that enforces one unit of grade A tomatoes to be used in paste (i.e., $AP \geq 1$).

	Name	Final Value	Reduced Cost	Obj Coeff	Upper Range	Lower Range
0	aw	525.0	0.000000	246.666667	709.777778	181.777778
1	aj	75.0	0.000000	198.000000	262.888889	-265.111111
2	ap	0.0	$\boxed{-97.333333}$	222.000000	319.333333	-inf
3	bw	175.0	0.000000	246.666667	1636.000000	181.777778
4	bj	225.0	0.000000	198.000000	240.962963	43.629630
5	bp	2000.0	0.000000	222.000000	inf	173.666667

The rest of the results are same (just shown in different order or different way).

	Name	Shadow Price	RHS Coeff	Slack	Upper Range	Lower Range
0	w_dem	0.000000	14400	13700.0	inf	700.000000
1	j_dem	0.000000	1000	700.0	inf	300.000000
2	p_dem	48.333333	2000	0.0	2200.000000	1533.333333
3	a_sup	271.000000	600	0.0	1200.000000	133.333333
4	b_sup	173.666667	2400	0.0	2866.666667	2200.000000
5	w_qual	-24.333333	0	0.0	466.666667	-600.000000
6	j_qual	-24.333333	0	0.0	1400.000000	-200.000000

9

9

Formulating a Product Blending Problem

- A manufacturer of plastics is planning to blend a new product by mixing four chemical compounds
- Each compound contains three chemicals A, B, and C in different percentages
- Table 1 gives for each compound its cost $/kg and the % of each chemical in it

Table 1	Comp 1	Comp 2	Comp 3	Comp 4
% of A	30	10	35	25
% of B	20	65	35	40
% of C	40	15	25	30
$/kg	20	30	20	30

- The new product must contain 25% of element A, at least 35% of element B, and at least 20% of element C
- Moreover, to avoid side effects compounds 1 and 2 cannot exceed 25% and 30% of the total, respectively
- Formulate a problem to solve what is the cheapest mix of compounds for blending one kg of the product?

10

10

Product Blending Formulation

Decision variables
x_i fraction of comp. i (i=1,…,4) used to produce 1 kg of product
(example: x_i=0.5 means that 1 kg of product has 50% of comp. i)

Objective function

Total cost to produce one kg of the new product

Constraints

% of comp. 1 + ⋯ + % of comp. 4 = 100%

exactly 25% of element A

at least 35% of element B

at least 20% of element C

at most 25% of comp. 1
at most 30% of comp. 2

non-negativity

11

Product Blending Formulation in Matrix Form

$$\begin{aligned} &\textit{Minimize } z = \mathbf{c}'\mathbf{x} \\ &\textit{s.t. } \mathbf{A}^1\mathbf{x} = \mathbf{b}^1 \\ &\quad \mathbf{A}^2\mathbf{x} \geq \mathbf{b}^2 \\ &\quad \mathbf{A}^3\mathbf{x} \leq \mathbf{b}^3 \\ &\quad \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where

$$\mathbf{A}^1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 30 & 10 & 35 & 25 \end{pmatrix} \quad \mathbf{b}^1 = \begin{pmatrix} 1 \\ 25 \end{pmatrix}$$

$$\mathbf{A}^2 = \begin{pmatrix} 20 & 65 & 35 & 40 \\ 40 & 15 & 25 & 30 \end{pmatrix} \quad \mathbf{b}^2 = \begin{pmatrix} 35 \\ 20 \end{pmatrix}$$

$$\mathbf{A}^3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{b}^3 = \begin{pmatrix} 0.25 \\ 0.30 \end{pmatrix}$$

$$\mathbf{c}' = (20 \quad 30 \quad 20 \quad 30)$$

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 30 & 10 & 35 & 25 \end{pmatrix}}_{\mathbf{A}^1} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} 1 \\ 25 \end{pmatrix}}_{\mathbf{b}^1}$$

$$\underbrace{\begin{pmatrix} 20 & 65 & 35 & 40 \\ 40 & 15 & 25 & 30 \end{pmatrix}}_{\mathbf{A}^2} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}_{\mathbf{x}} \geq \underbrace{\begin{pmatrix} 35 \\ 20 \end{pmatrix}}_{\mathbf{b}^2}$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{\mathbf{A}^3} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}_{\mathbf{x}} \leq \underbrace{\begin{pmatrix} 0.25 \\ 0.30 \end{pmatrix}}_{\mathbf{b}^3}$$

12

Product Blending Using Python and Gurobi

Decision variables

x_i fraction of comp. i (i=1,…,4) used to produce 1 kg of product (example: x_i=0.5 means that 1 kg of product has 50% of comp. i)

Objective function

$$minimize\ z = 20x_1 + 30x_2 + 20x_3 + 30x_4$$

Constraints

$$x_1 + x_2 + x_3 + x_4 = 1$$

$$30x_1 + 10x_2 + 35x_3 + 25x_4 = 25$$

$$20x_1 + 65x_2 + 35x_3 + 40x_4 \geq 35$$

$$40x_1 + 15x_2 + 25x_3 + 30x_4 \geq 20$$

$$x_1 \leq 0.25, x_2 \leq 0.30$$

$$x_1, x_2, x_3, x_4 \geq 0$$

```python
import gurobipy as gp
from gurobipy import GRB
import pandas as pd
from collections import OrderedDict

# Model
m = gp.Model("PB")

# Create decision variables for the fractions of compounds used
x1 = m.addVar(name="comp 1")
x2 = m.addVar(name="comp 2")
x3 = m.addVar(name="comp 3")
x4 = m.addVar(name="comp 4")

# The objective is to minimize the cost
obj = 20*x1+30*x2+20*x3+30*x4
m.setObjective(obj, GRB.MINIMIZE)

# % of compounds sums up to 100%
con1 = m.addConstr(x1+x2+x3+x4 == 1, name='com_sum')

# Exactly 25% of element A
con2 = m.addConstr(30*x1+10*x2+35*x3+25*x4 == 25, name='elem_a')

# At least 35% of element B
con3 = m.addConstr(20*x1+65*x2+35*x3+40*x4 >= 35, name='elem_b')

# At least 20% of element C
con4 = m.addConstr(40*x1+15*x2+25*x3+30*x4 >= 25, name='elem_c')

# At most 25% of comp1
con5 = m.addConstr(x1 <= 0.25, name='comp_1')

# At most 30% of comp 2
con6 = m.addConstr(x2 <= 0.30, name='comp_2')

# Non-negativity constraints
con7 = m.addConstr(x1>=0)
con8 = m.addConstr(x2>=0)
con9 = m.addConstr(x3>=0)
con10 = m.addConstr(x4>=0)

# Solve
m.optimize()
```

13

Product Blending Problem Output

```
Academic license - for non-commercial use only - expires 2022-07-19
Using license file C:\Users\jkettune\gurobi.lic
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 10 rows, 4 columns and 22 nonzeros
Model fingerprint: 0x141d213c
Coefficient statistics:
  Matrix range     [1e+00, 7e+01]
  Objective range  [2e+01, 3e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e-01, 4e+01]
Presolve removed 6 rows and 0 columns
Presolve time: 0.01s
Presolved: 4 rows, 4 columns, 16 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0000000e+01   8.500000e+00   0.000000e+00      0s
       2    2.4250000e+01   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds
Optimal objective  2.425000000e+01

Total cost to produce one kg of the new product: 24.25

Decision variables:
comp 1 = 0.25
comp 2 = 0.3
comp 3 = 0.325
comp 4 = 0.125


  Name  Final Value  Reduced Cost  Obj Coeff  Upper Range  Lower Range
0   x1        0.250           0.0         20         25.0         -inf
1   x2        0.300           0.0         30         45.0         -inf
2   x3        0.325           0.0         20         30.0         10.0
3   x4        0.125           0.0         30          inf         24.0


      Name  Shadow Price  RHS Coeff  Slack  Upper Range  Lower Range
0  com_sum          55.0       1.00  0.000        1.130     0.967647
1   elem_a          -1.0      25.00  0.000       26.250    21.750000
2   elem_b           0.0      35.00 -5.875       40.875         -inf
3   elem_c           0.0      25.00 -1.375       26.375         -inf
4   comp_1          -5.0       0.25  0.000        0.500     0.140000
5   comp_2         -15.0       0.30  0.000        0.350     0.083333
```

14

14

Summary

- After this session you should have:
  - obtained hands-on experience in working with Python and Gurobi
  - capability to interpret the outputs given by Python and Gurobi
- If you want to learn more about Python and Gurobi, you can watch videos at:
  - Python I: Introduction to Modeling with Python – Gurobi
  - (https://www.gurobi.com/resource/python-i-webinar/)

15

15