Workshop on Optimization Techniques for Data Science in Python and Julia

2. Solving linear programming (LP) problems with Pyomo

Bogumił Kamiński

Production planning
(Gueret et al., 2007, chap. 8.1)

A company produces bicycles. The following table presents the forecasted monthly demand for its product in thousands (`demand[i]`).

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
30	15	15	25	33	40	45	45	26	14	25	30

Normally company can produce 30 thousand bicycles per month. It can increase its production by overtime work by at most 50% per month. The cost to produce a bicycle normally is 32$, but it raises to 40$ in overtime.

The company can store produced bicycles in stock, which costs 5$ per bicycle charged at the end of the month. At the start of the year company has 2 thousand bicycles in stock.

Devise an optimal bicycle production plan for the company.

Solving optimization problems

1. Mathematical formulation
2. Problem type identification
3. Software implementation
4. Solution

Mathematical formulation

1. Decision variables

2. Objective

3. Constraints

Mathematical formulation

1. Decision variables
   - `prod_normal` 12 values of production levels in normal mode
   - `prod_overtime` 12 values of production levels in overtime mode
   - `store` 13 values of stock level at the end of the month
2. Objective
3. Constraints

Mathematical formulation

1. Decision variables
   - `prod_normal` 12 values of production levels in normal mode
   - `prod_overtime` 12 values of production levels in overtime mode
   - `store` 13 values of stock level at the end of the month
2. Objective
3. Constraints

Why 13?
Two types of variables:
flow and state

Mathematical formulation

1. Decision variables
   - `prod_normal` 12 values of production levels in normal mode
   - `prod_overtime` 12 values of production levels in overtime mode
   - `store` 13 values of stock level at the end of the month
2. Objective
   - minimize

$$\sum_{t=1}^{12} 32*\text{prod\_normal}[t]+40*\text{prod\_overtime}[t] +5*\text{store}[t]$$

3. Constraints

Mathematical formulation

1. Decision variables
    - `prod_normal` 12 values of production levels in normal mode
    - `prod_overtime` 12 values of production levels in overtime mode
    - `store` 13 values of stock level at the end of the month
2. Objective
    - minimize

$$\sum_{t=1}^{12} 32*\texttt{prod\_normal[t]}+40*\texttt{prod\_overtime[t]} +5*\texttt{store[t]}$$

3. Constraints
    - $0 \leq \texttt{prod\_normal[t]} \leq 30, 0 \leq \texttt{prod\_overtime[t]} \leq 15$
    - $0 \leq \texttt{store[t]}$, `store[0]=2`
    - `prod_normal[i]+prod_overtime[i]+store[i-1]=demand[i]+store[i]`

Problem type identification

- Decision variables: real
- Objective: linear in variables
- Constraints: linear in variables

Problem type identification

- Decision variables: real
- Objective: linear in variables
- Constraints: linear in variables

Type of problem:
linear programming

Software implementation (bikes.py)

```
from pyomo.environ import *

demand = [30.0, 15.0, 15.0, 25.0, 33.0, 40.0,
          45.0, 45.0, 26.0, 14.0, 25.0, 30.0]

cost_normal = 32 *1000
cost_overtime = 40 * 1000
cost_store = 5 * 1000
capacity = 30
n = len(demand)

model = ConcreteModel(name="Bicycle")
model.time = RangeSet(1,n)
model.store_time = RangeSet(0,n)
model.prod_normal = Var(model.time, bounds=(0,capacity))
model.prod_overtime = Var(model.time, bounds=(0,capacity / 2))
model.store = Var(model.store_time, within=NonNegativeReals)
```

Software implementation (bikes.py)

```
model.obj = Objective(expr=sum(cost_normal*model.prod_normal[i] +
                               cost_overtime*model.prod_overtime[i] +
                               cost_store*model.store[i] for i in model.time),
                               sense=minimize)

def time_constraint(model, t):
    inflow = model.prod_normal[t] + model.prod_overtime[t] + model.store[t-1]
    outflow = demand[t-1] + model.store[t]
    return inflow == outflow

model.constr = Constraint(model.time, rule=time_constraint)
model.store[0].fix(2.0)
```

Solution (bikes.py)

```
solver = SolverFactory("glpk")
results = solver.solve(model)
print(results)
model.pprint()

print("demand\tnormal\tover\tstore")
for t in range(1, 13):
    print("{:4}\t{:4}\t{:4}\t{:4}".format(demand[t-1],
          value(model.prod_normal[t]),
          value(model.prod_overtime[t]), value(model.store[t])))
```

Solution (bikes.py)

```
>>> print(results)

Problem:
- Name: unknown
  Lower bound: 11247000.0
  Upper bound: 11247000.0
  Number of objectives: 1
  Number of constraints: 13
  Number of variables: 37
  Number of nonzeros: 48
  Sense: minimize
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.04430723190307617
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Solution (bikes.py)

```
>>> model.pprint()
2 RangeSet Declarations
    store_time : Dim=0, Dimen=1, Size=13, Domain=Integers, Ordered=True, Bounds=(0, 12)
        Virtual
    time : Dim=0, Dimen=1, Size=12, Domain=Integers, Ordered=True, Bounds=(1, 12)
        Virtual

3 Var Declarations
    prod_normal : Size=12, Index=time
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          1 :     0 :  28.0 :    30 : False : False :  Reals
...
         12 :     0 :  30.0 :    30 : False : False :  Reals
...

1 Objective Declarations
    obj : Size=1, Index=None, Active=True
        Key  : Active : Sense    : Expression
        None :   True : minimize : 32000*prod_normal[1] + 40000*prod_overtime[1] + 5000*store[1] + 32000*prod_normal[2] +
40000*prod_overtime[2] +
...

1 Constraint Declarations
    constr : Size=12, Index=time, Active=True
        Key : Lower : Body                                                               : Upper : Active
          1 :   0.0 :     -30.0 + prod_normal[1] + prod_overtime[1] + store[0] - store[1] :   0.0 :   True
...
         12 :   0.0 : -30.0 + prod_normal[12] + prod_overtime[12] + store[11] - store[12] :   0.0 :   True

7 Declarations: time store_time prod_normal prod_overtime store obj constr
```

Solution (bikes.py)

```
>>> print("demand\tnormal\tover\tstore")
>>> for t in range(1, 13):
...     print("{:4}\t{:4}\t{:4}\t{:4}".format(demand[t-1],
...           value(model.prod_normal[t]),
...           value(model.prod_overtime[t]), value(model.store[t])))
...
demand  normal  over    store
30.0    28.0     0.0     0.0
15.0    15.0     0.0     0.0
15.0    15.0     0.0     0.0
25.0    28.0     0.0     3.0
33.0    30.0     0.0     0.0
40.0    30.0    10.0     0.0
45.0    30.0    15.0     0.0
45.0    30.0    15.0     0.0
26.0    26.0     0.0     0.0
14.0    14.0     0.0     0.0
25.0    25.0     0.0     0.0
30.0    30.0     0.0     0.0
```

Self-check task

What would change if the storage cost went down to 0?

What would change if the storage cost went up to 15?

Special situations (special.py)

- No feasible solution of the problem

$$\text{maximize} \quad x$$

s.t.

$$x \geq 0$$
$$x \leq -1$$

- Unbounded objective

$$\text{maximize } x$$

s.t.

$$x \geq 0$$
$$x \leq -1$$

Special situations (special.py)

- How to check solver status

```
results = solver.solve(model)
results.solver.termination_condition
```

- Most common solver status values

```
from pyomo.opt import TerminationCondition
TerminationCondition.optimal
TerminationCondition.unbounded
TerminationCondition.infeasible
```