Workshop on Optimization Techniques for Data Science in Python and Julia

4. Solving nonlinear programming problems with Pyomo

Bogumił Kamiński

Harvesting (Hart et al., chap. 7.4.2)

A population of deer is divided into: bucks, does and fawns.

Their number is modeled as a dynamical system in discrete time (time unit is one year).

It is assumed that each year hunters are allowed to harvest some number of bucks, does and fawns, where bucks are valued ten times more than does and fawns.

We want to maximize total value harvested in long-run.

Harvesting (Hart et al., chap. 7.4.2)

$$f_{t+1} = p_1 br_t \left(\frac{p_2}{10} f_t + p_3 d_t\right) - h_t^f$$

$$d_{t+1} = p_4 d_t + \frac{p_5}{2} f_t - h_t^d$$

$$b_{t+1} = p_6 b_t + \frac{p_5}{2} f_t - h_t^b$$

$$br_t = 1.1 + 0.8 \frac{p_s - c_t}{p_s}$$

$$c_t = p_7 b_t + p_8 d_t + p_9 f_t$$

- f_t, d_t, b_t: population of fawns, does and bucks in time t
- h_t^f, h_t^d, h_t^b: harvesting quota for fawns, does and bucks in time t
- br_t: birth rate of fawns in time t
- c_t: total food consumption in time t
- p_1 to p_9 and p_s: fixed parameters

Harvesting (Hart et al., chap. 7.4.2)

$$f_{t+1} = p_1 br_t\left(\frac{p_2}{10} f_t + p_3 d_t\right) - h_t^f$$
$$d_{t+1} = p_4 d_t + \frac{p_5}{2} f_t - h_t^d$$
$$b_{t+1} = p_6 b_t + \frac{p_5}{2} f_t - h_t^b$$
$$br_t = 1.1 + 0.8\frac{p_s - c_t}{p_s}$$
$$c_t = p_7 b_t + p_8 d_t + p_9 f_t$$

- f_t, d_t, b_t: population of fawns, does and bucks in time t
- h_t^f, h_t^d, h_t^b: harvesting quota for fawns, does and bucks in time t
- br_t: birth rate of fawns in time t
- c_t: total food consumption in time t
- p_1 to p_9 and p_s: fixed parameters

Harvesting (Hart et al., chap. 7.4.2)

$$f = p_1 br\left(\frac{p_2}{10}f + p_3 d\right) - h^f$$
$$d = p_4 d + \frac{p_5}{2}f - h^d$$
$$b = p_6 b + \frac{p_5}{2}f - h^b$$
$$br = 1.1 + 0.8\frac{p_s - c}{p_s}$$
$$c = p_7 b + p_8 d + p_9 f$$

- f, d, b: steady state population of fawns, does and bucks
- h^f, h^d, h^b: harvesting quota for fawns, does and bucks in steady state
- br: steady state birth rate of fawns
- c: steady state total food consumption
- p_1 to p_9 and p_s: fixed parameters

Harvesting (Hart et al., chap. 7.4.2)

- Parameters (estimated empirically):
  - $p_1 = 0.88$
  - $p_2 = 0.82$
  - $p_3 = 0.92$
  - $p_4 = 0.84$
  - $p_5 = 0.73$
  - $p_6 = 0.87$
  - $p_7 = 2700$
  - $p_8 = 2300$
  - $p_9 = 540$
  - $p_s = 700000$

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
    - $f, d, b, h^f, h^d, h^b, br, c$
2. Objective

3. Constraints

Mathematical formulation

1. Decision variables
    - $f, d, b, h^f, h^d, h^b, br, c$
2. Objective
    - maximize $h^f + h^d + 10h^b$
3. Constraints

Mathematical formulation

1. Decision variables
    - $f, d, b, h^f, h^d, h^b, br, c$
2. Objective
    - maximize $h^f + h^d + 10h^b$
3. Constraints
    - $f = p_1 br \left(\frac{p_2}{10} f + p_3 d\right) - h^f$
    - $d = p_4 d + \frac{p_5}{2} f - h^d$
    - $b = p_6 b + \frac{p_5}{2} f - h^b$
    - $br = 1.1 + 0.8 \frac{p_s - c}{p_s}$
    - $c = p_7 b + p_8 d + p_9 f \leq p_s$
    - $b \geq (0.4f + d)/5$
    - $f, d, b, h^f, h^d, h^b, br, c$ are non negative

Problem type identification

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

Problem type identification

- Decision variables: real
- Objective: non-linear (quadratic)
- Constraints: linear

maximize $-\frac{0.8p_1}{ps}(p_7 b + p_8 d + p_9 f)\left(\frac{p_2}{10} f + p_3 d\right)$+linear terms

s.t.

$$p_7 b + p_8 d + p_9 f \leq p_s$$

$f, d, b, h^f, h^d, h^b, br, c$ are non negative

Problem type identification

- Quadratic part

$$-\frac{0.8p_1}{ps}\begin{bmatrix} d & f & b \end{bmatrix}\begin{bmatrix} p_8p_3 & \frac{p_8p_2}{20}+\frac{p_9p_3}{2} & \frac{p_7p_3}{2} \\ \frac{p_8p_2}{20}+\frac{p_9p_3}{2} & \frac{p_9p_2}{10} & \frac{p_7p_2}{20} \\ \frac{p_7p_3}{2} & \frac{p_7p_2}{20} & 0 \end{bmatrix}\begin{bmatrix} d \\ f \\ b \end{bmatrix}$$

$$-1.006e-6\begin{bmatrix} d & f & b \end{bmatrix}\begin{bmatrix} 2116 & 342.7 & 1242 \\ 342.7 & 44.28 & 110.7 \\ 1242 & 110.7 & 0 \end{bmatrix}\begin{bmatrix} d \\ f \\ b \end{bmatrix}$$

is indefinite (so the problem is non-convex in general)

Problem type identification

Type of problem:
non linear programming
(possibly multiple local extrema)

- Quadratic part

$$-\frac{0.8p_1}{ps}\begin{bmatrix} d & f & b \end{bmatrix}\begin{bmatrix} p_8p_3 & \frac{p_8p_2}{20}+\frac{p_9p_3}{2} & \frac{p_7p_3}{2} \\ \frac{p_8p_2}{20}+\frac{p_9p_3}{2} & \frac{p_9p_2}{10} & \frac{p_7p_2}{20} \\ \frac{p_7p_3}{2} & \frac{p_7p_2}{20} & 0 \end{bmatrix}\begin{bmatrix} d \\ f \\ b \end{bmatrix}$$

$$-1.006e-6\begin{bmatrix} d & f & b \end{bmatrix}\begin{bmatrix} 2116 & 342.7 & 1242 \\ 342.7 & 44.28 & 110.7 \\ 1242 & 110.7 & 0 \end{bmatrix}\begin{bmatrix} d \\ f \\ b \end{bmatrix}$$

is indefinite (so the problem is non-convex in general)

Software implementation (harvesting.py)

```
from pyomo.environ import *

p1 = 0.88
p2 = 0.82
p3 = 0.92
p4 = 0.84
p5 = 0.73
p6 = 0.87
p7 = 2700.0
p8 = 2300.0
p9 = 540.0
ps = 700000.0

model = ConcreteModel(name="Harvesting")

model.f = Var(initialize = 20.0, within=NonNegativeReals)
model.d = Var(initialize = 20.0, within=NonNegativeReals)
model.b = Var(initialize = 20.0, within=NonNegativeReals)
model.hf = Var(initialize = 20.0, within=NonNegativeReals)
model.hd = Var(initialize = 20.0, within=NonNegativeReals)
model.hb = Var(initialize = 20.0, within=NonNegativeReals)
model.br = Var(initialize=1.5, within=NonNegativeReals)
model.c = Var(initialize=500000.0, within=NonNegativeReals)
```

Software implementation (harvesting.py)

```
def obj_rule(m):
    return 10*m.hb + m.hd + m.hf

model.obj = Objective(rule=obj_rule,
sense=maximize)

def f_bal_rule(m):
    return m.f == p1*m.br *(p2/10.0*m.f + p3*m.d)
- m.hf

model.f_bal = Constraint(rule=f_bal_rule)

def d_bal_rule(m):
    return m.d == p4*m.d + p5/2.0*m.f - m.hd

model.d_bal = Constraint(rule=d_bal_rule)

def b_bal_rule(m):
    return m.b == p6*m.b + p5/2.0*m.f - m.hb

model.b_bal = Constraint(rule=b_bal_rule)
```

```
def food_cons_rule(m):
    return m.c == p7*m.b + p8*m.d + p9*m.f

model.food_cons =
Constraint(rule=food_cons_rule)

def supply_rule(m):
    return m.c <= ps

model.supply = Constraint(rule=supply_rule)

def birth_rule(m):
    return m.br == 1.1 + 0.8*(ps - m.c)/ps

model.birth = Constraint(rule=birth_rule)

def minbuck_rule(m):
    return m.b >= 1.0/5.0*(0.4*m.f + m.d)

model.minbuck = Constraint(rule=minbuck_rule)
```

Solution (harvesting.py)

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

Solution (harvesting.py)

```
>>> print(results)

Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 7
  Number of variables: 8
  Sense: unknown
Solver:
- Status: ok
  Message: Ipopt 3.11.1\x3a Optimal Solution Found
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 0.0721292495727539
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Solution (harvesting.py)

```
>>> model.pprint()
8 Var Declarations
    b : Size=1, Index=None
        Key  : Lower : Value             : Upper : Fixed : Stale : Domain
        None :     0 : 54.36972761241992 :  None : False : False : NonNegativeReals
    br : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 1.0999999920111345 :  None : False : False : NonNegativeReals
    c : Size=1, Index=None
        Key  : Lower : Value             : Upper : Fixed : Stale : Domain
        None :     0 : 700000.0069902573 :  None : False : False : NonNegativeReals
    d : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 196.00640104188517 :  None : False : False : NonNegativeReals
    f : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 189.60559266738446 :  None : False : False : NonNegativeReals
    hb : Size=1, Index=None
        Key  : Lower : Value             : Upper : Fixed : Stale : Domain
        None :     0 : 62.13797673398074 :  None : False : False : NonNegativeReals
    hd : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 37.845017156893704 :  None : False : False : NonNegativeReals
    hf : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 :   0.0 :  None : False : False : PositiveReals

...

16 Declarations: f d b hf hd hb br c obj f_bal d_bal b_bal food_cons supply birth minbuck
```

Problem type identification: auxiliary analysis

1. Assume that $c < p_s$
2. Increase b by 1, then $10h^b$ goes up by 1.3
3. But this means that br goes down by < 0.0031
4. So h^f goes down by $< 0.000223696f + 0.000250976d$
5. But $\max(f, d) < 1300$, so h^f goes down by less than 0.62

Conclusion: in an optimal solution $c = p_s$ holds.

Type of auxiliary problem: linear programming

Software implementation (harvesting_lp.py)

```
from pyomo.environ import *

p1 = 0.88
p2 = 0.82
p3 = 0.92
p4 = 0.84
p5 = 0.73
p6 = 0.87
p7 = 2700.0
p8 = 2300.0
p9 = 540.0
ps = 700000.0
br = 1.1
```

```
model = ConcreteModel(name="Harvesting")

model.f = Var(within=NonNegativeReals)
model.d = Var(within=NonNegativeReals)
model.b = Var(within=NonNegativeReals)
model.hf = Var(within=NonNegativeReals)
model.hd = Var(within=NonNegativeReals)
model.hb = Var(within=NonNegativeReals)
model.c = Var(within=NonNegativeReals)

def obj_rule(m):
    return 10*m.hb + m.hd + m.hf

model.obj = Objective(rule=obj_rule, sense=maximize)

def f_bal_rule(m):
    return m.f == p1*br *(p2/10.0*m.f + p3*m.d) - m.hf
```

Software implementation (harvesting_lp.py)

```
from pyomo.environ import *

p1 = 0.88
p2 = 0.82
p3 = 0.92
p4 = 0.84
p5 = 0.73
p6 = 0.87
p7 = 2700.0
p8 = 2300.0
p9 = 540.0
ps = 700000.0
br = 1.1
```

Additionally changed solver to GLPK and removed lines:

```
def birth_rule(m):
    return m.br == 1.1 + 0.8*(ps - m.c)/ps

model.birth = Constraint(rule=birth_rule)
```

```
model = ConcreteModel(name="Harvesting")

model.f = Var(within=NonNegativeReals)
model.d = Var(within=NonNegativeReals)
model.b = Var(within=NonNegativeReals)
model.hf = Var(within=NonNegativeReals)
model.hd = Var(within=NonNegativeReals)
model.hb = Var(within=NonNegativeReals)
model.c = Var(within=NonNegativeReals)

def obj_rule(m):
    return 10*m.hb + m.hd + m.hf

model.obj = Objective(rule=obj_rule, sense=maximize)

def f_bal_rule(m):
    return m.f == p1*br *(p2/10.0*m.f + p3*m.d) - m.hf
```

Solution (harvesting.py)

```
7 Var Declarations
    b : Size=1, Index=None
        Key  : Lower : Value            : Upper : Fixed : Stale : Domain
        None :     0 : 54.3697271084899 :  None : False : False : NonNegativeReals
    c : Size=1, Index=None
        Key  : Lower : Value    : Upper : Fixed : Stale : Domain
        None :     0 : 700000.0 :  None : False : False : NonNegativeReals
    d : Size=1, Index=None
        Key  : Lower : Value            : Upper : Fixed : Stale : Domain
        None :     0 : 196.006398762916 :  None : False : False : NonNegativeReals
    f : Size=1, Index=None
        Key  : Lower : Value            : Upper : Fixed : Stale : Domain
        None :     0 : 189.605591948833 :  None : False : False : NonNegativeReals
    hb : Size=1, Index=None
        Key  : Lower : Value            : Upper : Fixed : Stale : Domain
        None :     0 : 62.1379765372205 :  None : False : False : NonNegativeReals
    hd : Size=1, Index=None
        Key  : Lower : Value            : Upper : Fixed : Stale : Domain
        None :     0 : 37.8450172592576 :  None : False : False : NonNegativeReals
    hf : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 :   0.0 :  None : False : False : NonNegativeReals

...

14 Declarations: f d b hf hd hb c obj f_bal d_bal b_bal food_cons supply minbuck
```

Self-check task

Chceck what would happen if we removed the condition on minimum number of bucks.

Can you explain the result?