UNIVERSITÄT LEIPZIG

Energiemanagement und Nachhaltigkeit

Pyomo – Optimierung mit Python

Strommarkt-TechTalk – Berlin – 20.02.2015

Philipp Hanemann

Professur für Energiemanagement und Nachhaltigkeit
Institut für Infrastruktur und Ressourcenmanagement - IIRM
Universität Leipzig

Warum Python?

- Open source
- Vollwertige Programmiersprache (high level)
- Vektorrechnung (Numpy, Scipy)
- Zeitreihen (Pandas)
- **Optimierung** (cvx-opt, Pyomo)
- viele andere Libraries
- große Community → dynamische Entwicklung

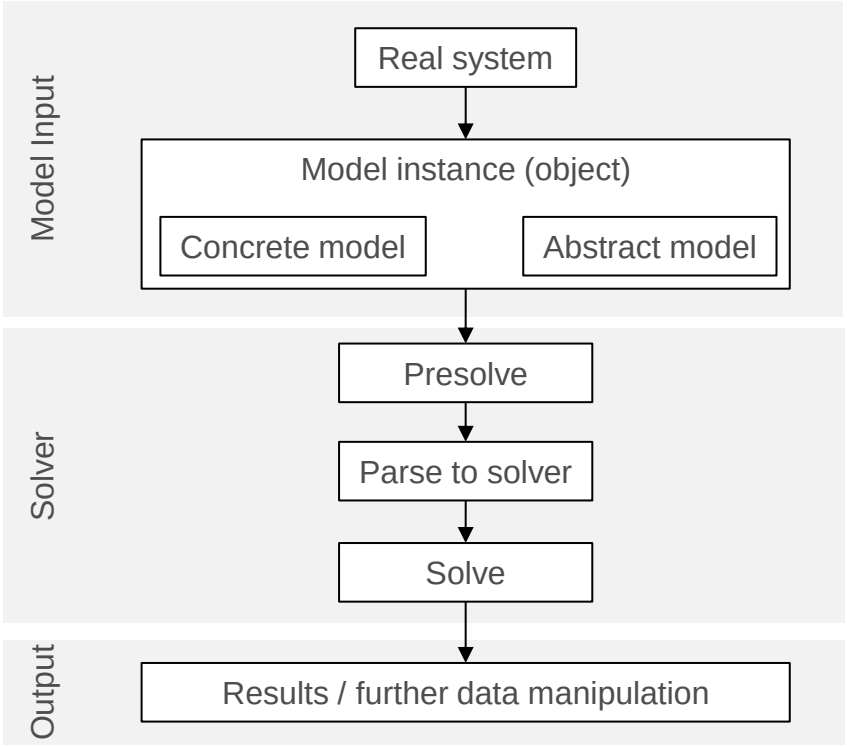
Ein Tool für alles

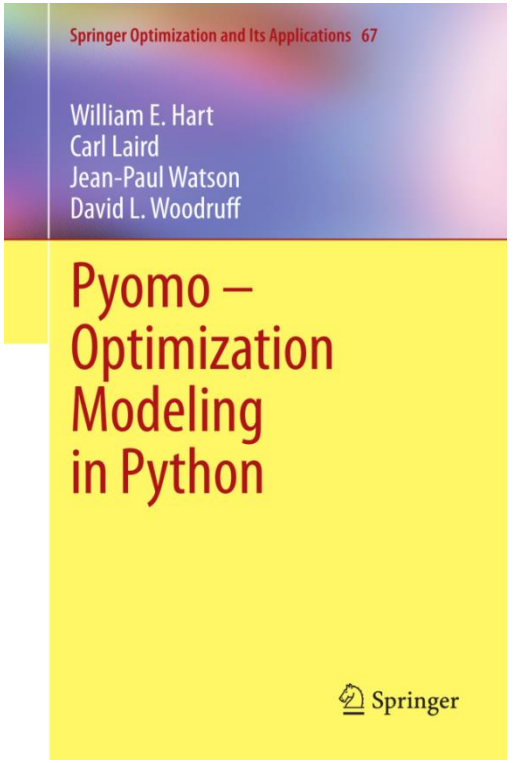
Pyomo (http://www.pyomo.org/ - new ☺)

„The goal of Pyomo is to provide a platform for expressing optimization models that supports the central ideas of modern **algebraic modeling language**s within a framework that promotes *flexibility*, *extensibility*, *portability*, and *maintainability*."

Latest version: 4.0 (22nd January 2015)

In Python everything is an object!!!





Concrete Model

Problem

max $3x_1 + 5x_2$

s.t.:

$$x_1 \leq 4$$
$$x_2 \leq 6$$
$$3x_1 + 2x_2 \leq 18$$
$$x_1 \geq 0, x_2 \geq 0$$

```
from coopr.pyomo import *
from coopr.opt import SolverFactory

m = ConcreteModel()

m.x1 = Var()
m.x2 = Var()
m.obj = Objective(expr = 3*m.x1 + 5*m.x2,
                  sense = maximize)
m.con1 = Constraint(expr = 1*m.x1 + 0*m.x2 <= 4)
m.con2 = Constraint(expr = 0*m.x1 + 1*m.x2 <= 6)
m.con3 = Constraint(expr = 3*m.x1 + 2*m.x2 <= 18)
m.con4 = Constraint(expr = m.x1 >= 0)
m.con5 = Constraint(expr = m.x2 >= 0)


inst = m.create()


opt = SolverFactory("glpk")
results = opt.solve(inst)
```

Pyomo library

Model object

Model specifciation

Model instance

Solver specification

Model solve

Abstract Model

Problem

max $3x_1 + 5x_2$

s.t.:
$$x_1 \leq 4$$
$$x_2 \leq 6$$
$$3x_1 + 2x_2 \leq 18$$
$$x_1 \geq 0, x_2 \geq 0$$

$\max\{c^T x \mid Ax \leq b, x \geq 0\}$
with $A \epsilon \mathbb{R}^{m,n}, b \epsilon \mathbb{R}^m, c \epsilon \mathbb{R}^n$

```
from coopr.pyomo import *
from coopr.opt import SolverFactory

#instanciating a mel
m = AbstractModel()
m.V = Set()
m.R = Set()
m.c = Param(m.V)
m.a = Param(m.R, m.V)
m.b = Param(m.R)
m.x = Var(m.V, within = NonNegativeReals)

def obj_rule(m):
    return sum(m.c[i]*m.x[i] for i in m.V)
m.obj = Objective(rule=obj_rule, sense = maximize)

def con_rule(m, r):
    return sum(m.a[r,i] * m.x[i] for i in m.V) <= m.b[r]
m.con = Constraint(m.R, rule=con_rule)

instance = m.create('lp_1_abstract_input.dat')

opt = SolverFactory("glpk")
results = opt.solve(instance)
```

Sets: *V* columns, *R* rows

Parameters for: *c*, *A*, *b*

$$obj = \sum_{i \in V} c_i * x_i$$

$$\sum_{i \in V} a_i * x_i \leq b_r \quad \forall r \in R$$

Input data

Kontakt

Philipp Hanemann

Professur für Energiemanagement und Nachhaltigkeit

Wirtschaftswissenschaftliche Fakultät
Universität Leipzig
Grimmaische Str. 12
D-04109 Leipzig

Tel.: 0341/97 33858

philipp.hanemann@uni-leipzig.de

www.wifa.uni-leipzig.de/iirm