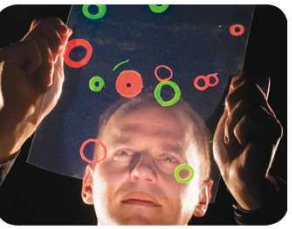
SAND2015-0059C



Sandia National Laboratories

Exceptional service in the national interest

Modeling Bilevel Programs in Pyomo

Richard L. Chen
William E. Hart
John D. Siirola
Jean-Paul Watson
Sandia National Laboratories

January 11, 2014

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Overview of Bilevel Programming

A *bilevel program* is a mathematical program in which a subset of decision variables is constrained to take values associated with an optimal solution of a distinct, "lower" level mathematical program.

General formulation:

$$\begin{array}{ll} \min_{x \in X} & F(x, y) \\ \text{s.t.} & G(x, y) \leq 0 \\ & y \in P(x) \end{array}$$

Upper-level problem

where

$$P(x) = \begin{array}{ll} \operatorname{argmin}_{y \in Y} & f(x, y) \\ \text{s.t.} & g(x, y) \leq 0 \end{array}$$

Lower-level problem

Example: Modeling Security Problems

- Opponents must anticipate each other's moves
- Strategy should account for how opponent (best) responds

Extremely complex

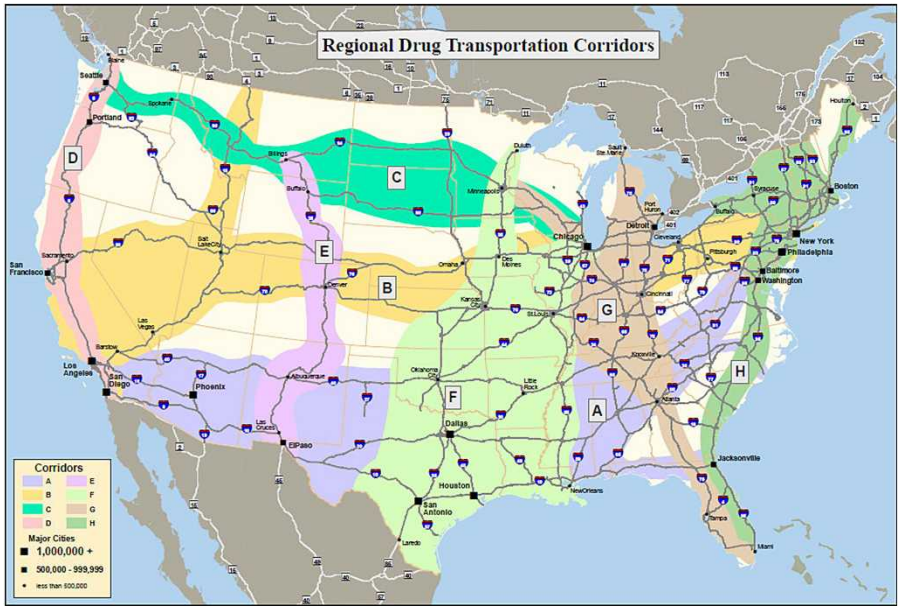
- Impossible to enumerate the set of all states in the game

Stackelberg games – bilevel programs

Example: Smuggling Interdiction

Interdictor minimizes the potential for a smuggler to evade detection

- Interdictor installs defenses (x) to minimize smuggler's evasion probability
- Smuggler traverses path (y) that maximizes the probability of evasion



Origin-destination nodes maybe unknown

- *F(x,y)* – evasion probability
- *X* – interdictor's constraints (e.g. resource, budget)
- *P(x)* – feasible paths given *x*

Modeling Bilevel Programs

No algebraic modeling language currently provides an *intuitive* syntax for expressing the structure of bilevel programs!

MPEC formulations are supported in several AMLs

- AMPL, AIMMS, GAMS, …
- MacMPEC includes bilevel programs that are reformulated as single-level programs using optimality conditions

Bilevel problems can be expressed in several modeling languages:

- GAMS, YALMIP
- Explicitly pass variables and constraints to a bilevel solver

A Simple Bilevel Example

Practical Bilevel Optimization: Algorithms and Applications

Jonathan Bard

Example 5.1.1

$$
\begin{array}{lll}
\min_{x \geq 0} & x - 4y & \\
\text{s.t.} & \min_{y \geq 0} & y \\
 & \text{s.t.} & -x - y \leq -3 \\
 & & -2x + y \leq 0 \\
 & & 2x + y \leq 12 \\
 & & -3x + 2y \leq -4
\end{array}
$$

Modeling Example 5.1.1 with YALMIP

```
sdpvar x, y;

OO = x – 4y;
CO = [x] >= 0;

OI = y;
CI = [[y] >= 0,
      -x -  y <= -3,
     -2x +  y <=  0,
      2x +  y <= 12,
     -3x + 2y <= -4];

solvebilevel(CO,OO,CI,OI,[y])
```

- Solve a bilevel problem using a simple branching strategy.
- Upper level problem defined by CO and OO
- Lower level problem defined by CI and OI, with decision variable y.

Note: This example adapted from the YALMIP bilevel documentation.

Modeling Example 5.1.1 with GAMS

```
positive variables x,y; variables objout,objin;
equations defout,defin,e1,e2,e3,e4;

defout.. objout =e= x - 4*y;
defin..  objin  =e= y;

e1..    - x -   y =l= -3;
e2..   -2*x +   y =l=  0;
e3..    2*x +   y =l= 12;
e4..    3*x - 2*y =l=  4;

model bard / all /;

$echo bilevel x min objin * defin e1 e2 e3 e4 > "%emp.info%"

solve bard us emp min objout;
```

Writes an "empinfo" file that tells the solver that this is a bilevel problem with a lower level problem that minimizes objective *objin* with variables *y* subject to the constraints (*defin*), *e1*, *e2*, *e3* and *e4*.

Modeling Example 5.1.1 with Pyomo

```
from pyomo.environ import *
from pyomo.bilevel import *

M = ConcreteModel()
M.x = Var(bounds=(0,None))
M.y = Var(bounds=(0,None))
M.o = Objective(expr=M.x - 4*M.y)

M.sub = SubModel(fixed=M.x)
M.sub.o = Objective(expr=M.y)
M.sub.c1 = Constraint(expr=-  M.x -   M.y <= -3)
M.sub.c2 = Constraint(expr=-2*M.x +   M.y <=  0)
M.sub.c3 = Constraint(expr= 2*M.x +   M.y <= 12)
M.sub.c4 = Constraint(expr=-3*M.x + 2*M.y <= -4)
```

- Lower level problem is declared with a *SubModel* component.
- The *var* argument indicates the lower level variables.
- Objectives, variables and constraints for the lower level problem are declared within this component.

Pyomo Extensions for Bilevel Programs

Modeling extensions

- Modeling components (pyomo.bilevel)

Model transformations

- Can be applied automatically

Custom solvers

- *Solvers tailored for specific classes of bilevel problems*

Solving Bilevel Problems

Goal: Enable solution of bilevel problems with standard solvers

Process:

- Model problem with SubModel components
- Transform the problem to a standard form
  - LP, MIP, etc
- Apply a suitable solver

Reformulations for linear bilevel programming (BLP)

A. BLP with continuous variables
B. Quadratic minimax with continuous lower-level variables

(A) BLP with Continuous Variables

Problem:

$$\begin{aligned} \min_{x \geq 0} \quad & c_1^T x + d_1^T y \\ \text{s.t.} \quad & A_1 x + B_1 y \leq b_1 \\ & \min_{y \geq 0} \quad c_2^T x + d_2^T y \\ & \qquad\qquad A_2 x + B_2 y \leq b_2 \end{aligned}$$

Reformulation: Replace lower-level problem with corresponding optimality conditions

$$\begin{aligned} \min \quad & c_1^T x + d_1^T y \\ \text{s.t.} \quad & A_1 x + B_1 y \leq b_1 \\ & d_2 + B_2^T u - v = 0 \\ & b_2 - A_2 x - B_2 y \geq 0 \perp u \geq 0 \\ & y \geq 0 \perp v \geq 0 \\ & x \geq 0, y \geq 0 \end{aligned}$$

(A) BLP with Continuous Variables (cont'd)

Idea: Analyze the MPEC reformulation

Example:

- Use a custom solver that considers complementarity conditions (Bard, 1998)

Example:

- Chain reformulations: BLP -> MPEC -> GDP -> MIP
- Provide "BigM" values for unbounded variables
- Apply standard MIP solver (Fortuny-Amat and McCarl, 1981)

Example:

- Reformulate the complementarity conditions with nonlinear constraints (Ferris and Dirkse, 2005)

(B) Quadratic Min/Max

Problem:

- Upper level constraints do not constrain *y*

$$X = \{x | A_1 x \leq b_1, x \geq 0\}$$

- The upper decision variables may binary

$$\begin{aligned} \min_{x \in X} \quad \max_{y \geq 0} \quad & c_1^T x + d_1^T y + x^T Q y \\ \text{s.t.} \quad & A_2 x + B_2 y \leq b_2 \end{aligned}$$

Reformulation: Replace lower-level problem with the linear dual

$$\begin{aligned} \min \quad & c_1^T x + (b_2 - A_2 x)^T v \\ \text{s.t.} \quad & B_2^T v \geq d_1 + Q^T x \\ & A_1 x \leq b_1 \\ & x \geq 0, v \geq 0 \end{aligned}$$

(B) Quadratic Min/Max (cont'd)

Case 1:

- $A_2 \equiv 0$
- The reformulation is a simple LP (or a MIP if x are binary)

Case 2:

- The upper-level decision variables *x* are binary
- Reformulate the bilinear objective terms as disjunctions:

$$
\begin{aligned}
\min \quad & c_1^T x + b_2^T v - 1^T z \\
\text{s.t.} \quad & B_2^T v \geq d_1 + Q^T x \\
& \begin{bmatrix} x_i = 0 \\ z_i = 0 \end{bmatrix} \veebar \begin{bmatrix} x_i = 1 \\ z_i = A_2^T(i,*)v \end{bmatrix} \\
& x \in X, v \geq 0
\end{aligned}
$$

- Reformulate this GDP -> MIP using "BigM" values

Solving Bilevel Programs in Pyomo

Python Script:

- Formulate the model
- Apply desired model reformulations
- Apply a suitable optimizer

  OR
- Directly analyze the model within Python
  - (e.g. using Pyomo's algebraic structure)

Pyomo Command:

- Execute a command that executes a Pyomo meta-solvers
  - Performs suitable reformulations
  - Applies a suitable optimizer
  - Maps the solution to the original problem

```
pyomo solve --solver=bilevel_ld model.py
```

Pyomo Capabilities

Relevant Pyomo Transformations

- core.linear_dual
- bilevel.linear_dual
- bilevel.linear_mpec
- gdp.bigm
- gdp.bilinear
- gdp.chull
- mpec.simple_disjunction
- mpec.simple_nonlinear

Relevant Pyomo Meta-Solvers

- bilevel_ld

Ongoing Work …

- Generalization and maturation of transformations
  - E.g. Working with general BLP models
- Automatic recognition of bilevel structure
  - Can we automate the application of reformulations?
- Parameterizing transformations
  - How can we flexibly specify transformation options?
  - E.g. Specifying big-M values for specific complementarity conditions
- Additional meta-solvers
  - E.g. bilevel_blp

For More Information

See the new Pyomo homepage

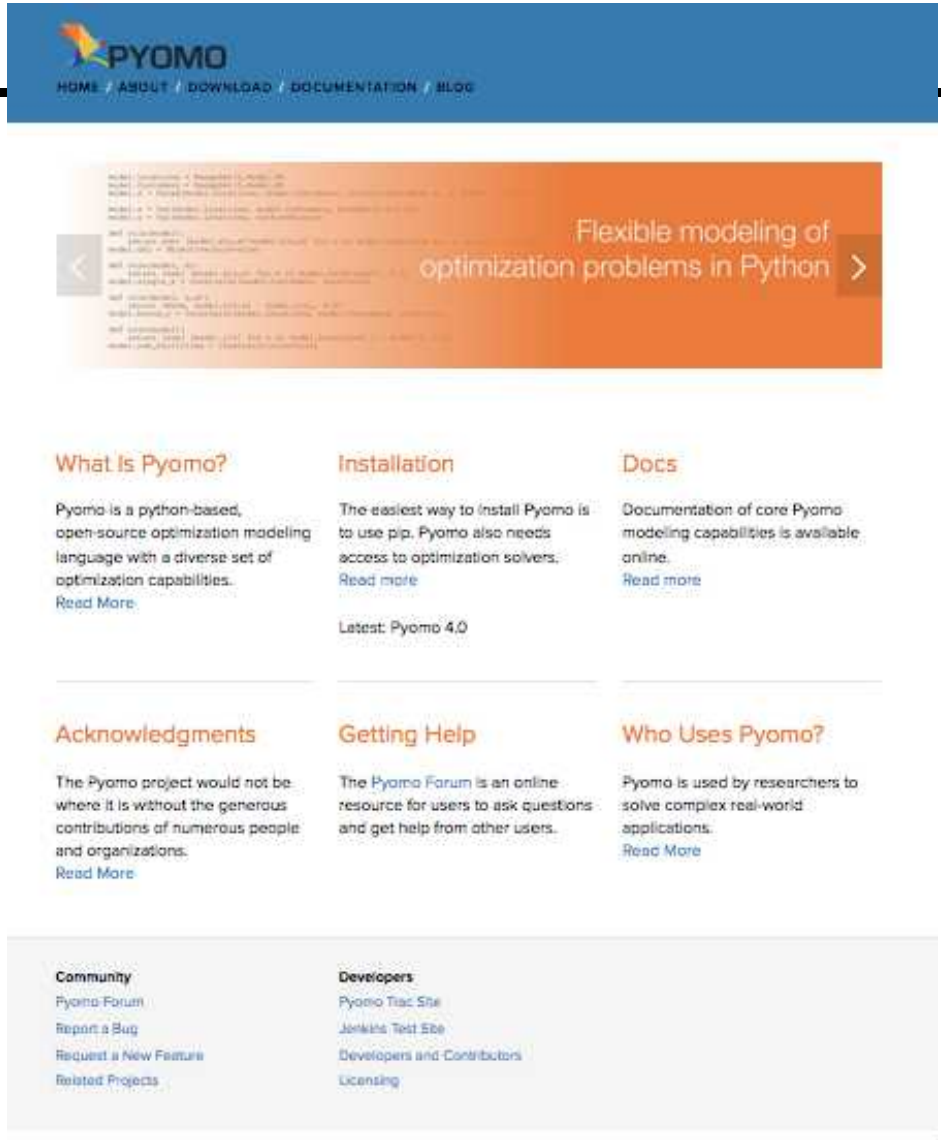
- www.pyomo.org

The Pyomo hompage provides a portal for:

- Online documentation
- Installation instructions
- Help information
- Developer links

Coming soon:

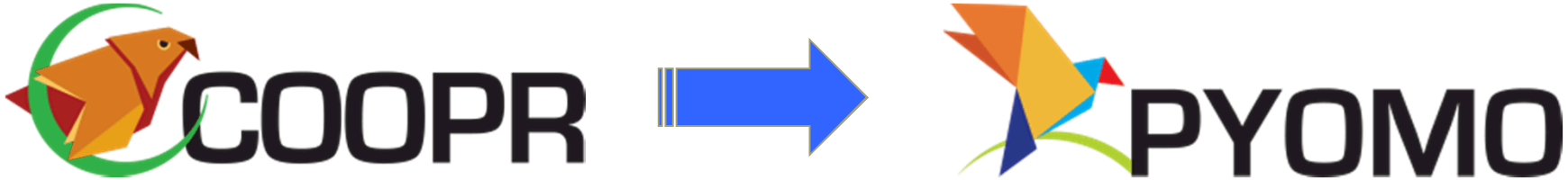
- Blogging about Pyomo capabilities and features
- A gallery of simple examples



The End

Questions?

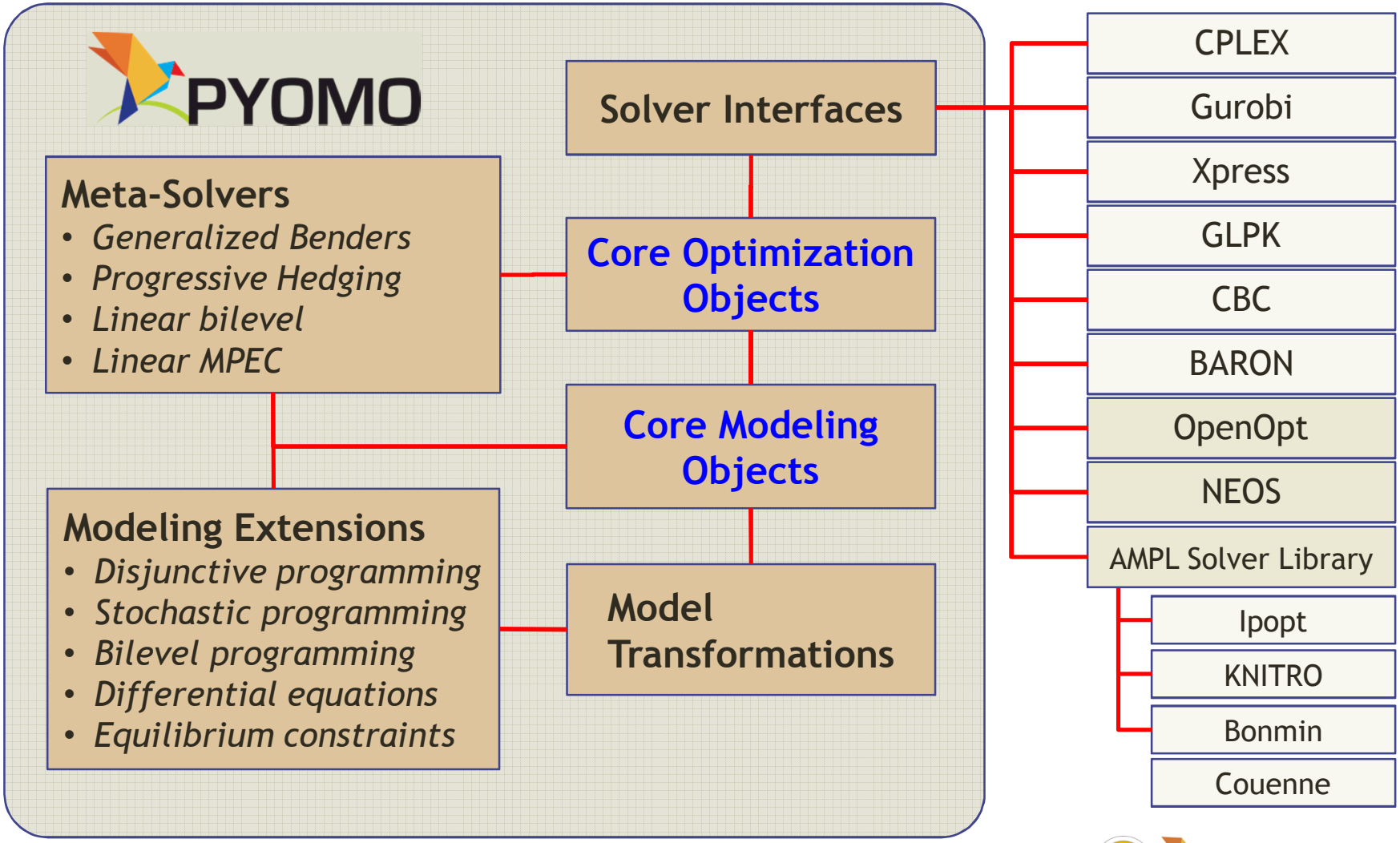
What Happened to Coopr?



- Users were installing Coopr but using Pyomo
  - Pyomo modeling extensions were not distinct enough
  - Researchers cited “Coopr/Pyomo”
- Users/Developers were confused by the coopr and pyomo commands
- Developers were coding in Coopr but talking about Pyomo

We need to provide clear branding this project!

Pyomo at a Glance



(A) BLP with Continuous Variables

Problem:

$$\begin{aligned} \min_{x \geq 0} \quad & c_1^T x + d_1^T y \\ \text{s.t.} \quad & A_1 x + B_1 y \leq b_1 \\ & \min_{y \geq 0} \quad c_2^T x + d_2^T y \\ & \qquad\qquad A_2 x + B_2 y \leq b_2 \end{aligned}$$

Reformulation: This is the MPEC model that eliminates the v variable in the reformulation on the earlier slide.

$$\begin{aligned} \min \quad & c_1^T x + d_1^T y \\ \text{s.t.} \quad & A_1 x + B_1 y \leq b_1 \\ & b_2 - A_2 x - B_2 y \geq 0 \perp u \geq 0 \\ & y \geq 0 \perp d_2 + B_2^T u \geq 0 \\ & x \geq 0, y \geq 0 \end{aligned}$$