SAND 2014-2616C

Analyzing Structured Optimization Models with Automatic Transformations

William E. Hart and John D. Siirola

Analytics Department
Sandia National Laboratories
Albuquerque, NM USA

{wehart,jdsiiro}@sandia.gov

June 4, 2014

MINLP Workshop – CMU

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.

Is this an *optimization model*?

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \Re^n \end{array}$$

Models are for *Modelers*

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \le b \\ & x \in \Re^n \end{array}$$

- I would argue this is an *optimization problem!*
- So, what's a *model*?
  - A general representation of a class of problems
    - Data (instance) independent
  - Represents the modeler's understanding of the class of problems
    - Explicitly annotates and conveys the class structure
  - Incorporates assumptions and simplifications
  - Is both tractable and valid
    - (although these are often contradictory goals)

Models are for *Modelers*

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x \in \Re^n \end{aligned}$$

- I would argue this is an *optimization problem!*
- So, what's a *model*?
  - A general representation of a class of problems
    - Data (instance) independent
  - Represents the modeler's understanding of the class of problems
    - Explicitly annotates and conveys the class structure
  - Incorporates assumptions and simplifications
  - Is both tractable and valid
    - (although these are often contradictory goals)

Optimization problems: Model instances

Model + Data → Problem

- We seldom have a single *problem* to solve
  - Rather we would like to write a *single model* for a *class of problems*
    - Key design feature of many AMLs (e.g. strongly encouraged by AMPL)
  - Why?
    - Test small, deploy big
    - Tomorrow's problem is different from today's
    - Data may be
      - Huge
      - Machine-generated
      - Stored externally (loaded from external tools, e.g. databases)

Models are for *Modelers*

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \Re^n \end{array}$$

- I would argue this is an *optimization problem!*
- So, what's a *model*?
  - A general representation of a class of problems
    - Data (instance) independent
  - Represents the modeler's understanding of the class of problems
    - Explicitly annotates and conveys the class structure
  - Incorporates assumptions and simplifications
  - Is both tractable and valid
    - (although these are often contradictory goals)

Sandia National Laboratories

What is *model structure*?

$$\begin{array}{ll} \min & c^T x \\ s.t. & \color{red}{A}x \le b \\ & x \in \Re^n \end{array}$$

- Unlike a solver, modelers don't think in terms of "A"
  - Rather, I think in terms of repeated (indexed) units
    - Sets (1-, 2-, n- dimensional)
    - Vectors or matrices of variables
    - Groups of related constraints (blocks)
- The model may not be "flat"
  - Block diagonal (e.g., scenarios in stochastic programming)
  - Graph-based (e.g., network flow)
  - Hierarchically defined (e.g., a model composed of sub-models)

Models are for *Modelers*

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \Re^n \end{array}$$

- I would argue this is an *optimization problem!*
- So, what's a *model*?
  - A general representation of a class of problems
    - Data (instance) independent
  - Represents the modeler's understanding of the class of problems
    - Explicitly annotates and conveys the class structure
  - Incorporates assumptions and simplifications
  - Is both tractable and valid
    - (although these are often contradictory goals)

Tractability / validity: The optimization tug-of-war

- The "highest fidelity" model of a system is rarely tractable
  - Delicate balance between the model we want to solve and the solver we want to use
  - What can we do?
    - Simplify (reduce the model scope)
    - Approximate (relax or recast constraints)
    - Iterate (solve a series of related problems to develop the solution to the original problem)
  - Optimization 101 ingrains this tension into us; consider:

$$\begin{array}{ll} \max & abs(x-3) \\ s.t. & [\ldots] \end{array}$$

"Modeling" absolute value

- This probably makes you cringe:
  - "Experienced modelers would never write `abs()`!"

$$\begin{aligned} \max \quad & abs(x-3) \\ s.t. \quad & [\ldots] \end{aligned}$$

- Instead, we write:

$$\begin{aligned} \max \quad & absX \\ s.t. \quad & absX = negX + posX \\ & negX \le My \\ & posX \le M(1-y) \\ & X-3 = posX - negX \\ & posX \ge 0, negX \ge 0 \\ & y \in \{0,1\} \\ & [\ldots] \end{aligned}$$

"Modeling" absolute value

- This probably makes you cringe:
  - "Experienced modelers would never write abs()!"

$$\begin{array}{ll} \max & abs(x-3) \\ s.t. & [\ldots] \end{array}$$

- Instead, we write:

$$\begin{array}{ll} \max & absX \\ s.t. & absX = negX + posX \\ & negX \le My \\ & posX \le M(1-y) \\ & X - 3 = posX - negX \\ & posX \ge 0, negX \ge 0 \\ & y \in \{0,1\} \\ & [\ldots] \end{array}$$

- But what if "[…]" is a nonlinear model? Then,

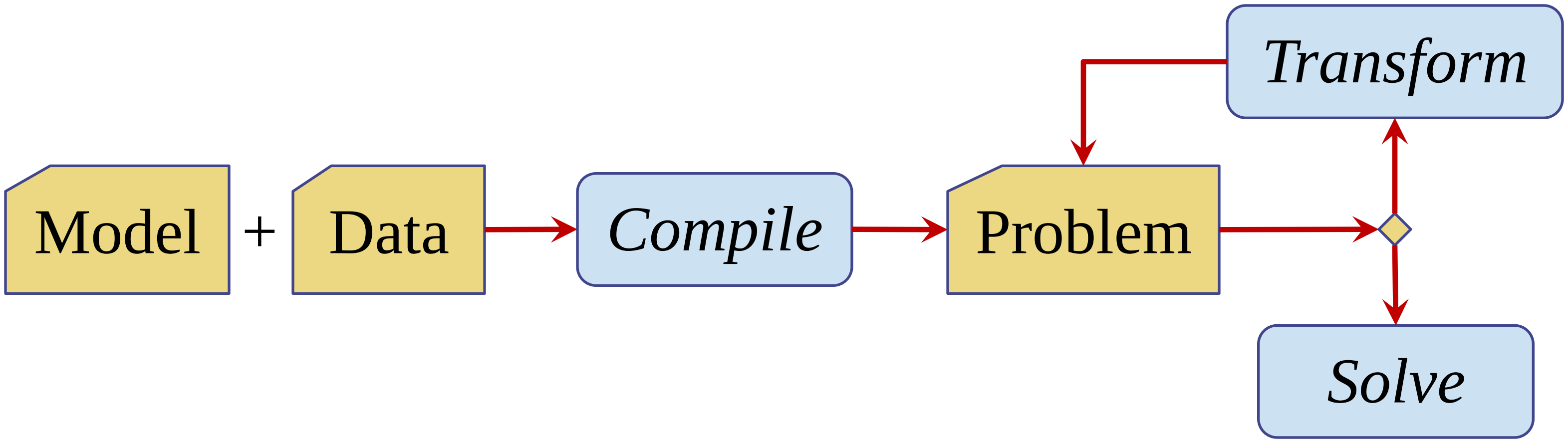
$$absX = \sqrt{x^2 + \varepsilon}$$

$$absX = \frac{2x}{1+e^{-x/h}} - x$$

- Does any of this really encode our understanding of the *class of problems*?
  - …or is this a reflection of our understanding of the *solver*?

Transformations: *Projecting problems to problems*

- Model Transformations
  - Project from one problem space to another
  - Standardize common reformulations or approximations
  - Convert “unoptimizable” modeling constructs into equivalent optimizable forms



Sandia National Laboratories

Transformations are not entirely new

- LINGO's automatic linearization:

```
MODEL:
  MAX = @ABS( X-3 );
  X <= 2;
END
```

  - Generates the "usual" Big-M integer linear model:

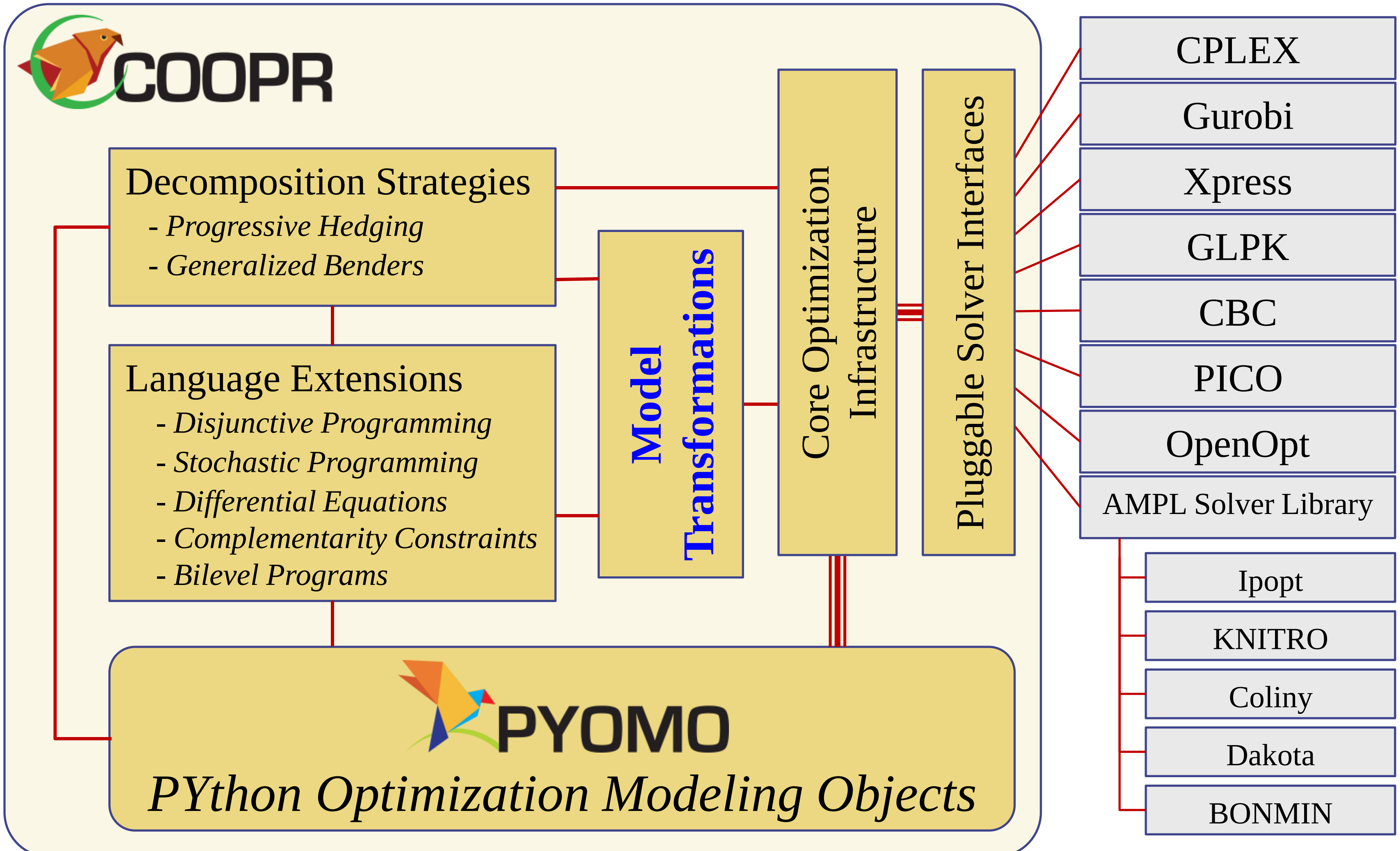
```
MAX _C3
SUBJECT TO
  X <= 2
  - _C1 - _C2 + _C3 = 0
  _C1 – 100000 _C4 <= 0
  _C2 + 100000 _C4 <= 100000
  X - _C1 + _C2 = 3
END
INTE _C4
```

Cunningham and Schrage, "The LINGO Algebraic Modeling Language." In *Modeling Languages in Mathematical Optimization*, Josef Kallrath ed. Springer, 2004.

Why are we interested in transformations?

- Separate model expression from how we intend to solve it
  - Defer decisions that improve tractability until solution time
  - Explore alternative reformulations or representations
  - Support *solver-specific* model customizations (e.g., `abs()`)
  - Support iterative methods that use different solvers requiring different representations (e.g., initializing NLP from MIP)
- Support "higher level" or non-algebraic modeling constructs
  - Express models that are closer to reality, e.g.:
    - Piecewise expressions
    - Disjunctive models (switching decisions & logic models)
    - Differential-algebraic models (dynamic models)
    - Bilevel models (game theory models)
- Reduce "mechanical" errors due to manual transformation

Coopr: *a COmmon Optimization Python Repository*



Sandia National Laboratories

A Quick Tour of Pyomo

Idea: a Pythonic framework for formulating optimization models

- Provides a natural syntax to describe mathematical models
- Leverages an extensible optimization object model
- Formulates large models with a concise syntax
- Separates modeling and data declarations
- Enables data import and export in commonly used formats

Highlights:

- Python provides a clean, readable syntax
- Python scripts provide a flexible context for exploring the structure of Pyomo models

```
from coopr.pyomo import *

model = ConcreteModel()

model.x1 = Var()
model.x2 = Var(bounds=(-1,1))
model.x3 = Var(bounds=(1,2))

model.obj = Objective(
    expr= m.x1**2 + (m.x2*m.x3)**4 +
          m.x2*sin(m.x1+m.x3) + m.x2,
    sense= minimize)
```

Structural transformations: Disjunctive programs

- Disjunctions: selectively enforce sets of constraints
  - Sequencing decisions: x ends before y or y ends before x
  - Switching decisions: a process unit is built or not
  - Alternative selection: selecting from a set of pricing policies

- Implementation: leverage Pyomo blocks
  - `Disjunct`:
    - Block of Pyomo components
      - (Var, Param, Constraint, etc.)
    - Boolean (binary) indicator variable determines if block is enforced
  - `Disjunction`:
    - Enforces logical XOR across a set of Disjunct indicator variables
  - (Logic constraints on indicator variables)

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le o \\ c_k = \gamma_{ik} \end{bmatrix}$$

$$\Omega(Y) = true$$

Example: Task sequencing

- Prevent tasks colliding on a single piece of equipment
  - Derived from Raman & Grossmann (1994)
  - Given:
    - Tasks I processed on a sequence of machines (with no waiting)
    - Task i starts processing at time t_i with duration τ_{im} on machine m
    - $J(i)$ is the set of machines used by task i
    - C_{ik} is the set of machines used by both tasks i and j

$$\begin{bmatrix} Y_{ik} \\ t_i + \sum_{\substack{m \in J(i) \\ m \le j}} \tau_{im} \le t_k + \sum_{\substack{m \in J(k) \\ m < j}} \tau_{km} \end{bmatrix} \vee \begin{bmatrix} Y_{ki} \\ t_k + \sum_{\substack{m \in J(k) \\ m \le j}} \tau_{km} \le t_i + \sum_{\substack{m \in J(i) \\ m < j}} \tau_{im} \end{bmatrix}$$

$$\forall j \in C_{ik}, \forall i, k \in I, i < k$$

Sandia National Laboratories

Example: Task sequencing in Coopr

```
def _NoCollision(model, disjunct, i, k, j, ik):
    lhs = model.t[i] + sum(model.tau[i,m] for m in model.STAGES if m<j)
    rhs = model.t[k] + sum(model.tau[k,m] for m in model.STAGES if m<j)
    if ik:
        disjunct.c = Constraint( expr= lhs + model.tau[i,j] <= rhs )
    else:
        disjunct.c = Constraint( expr= rhs + model.tau[k,j] <= lhs )
model.NoCollision = Disjunct( model.L, [0,1], rule=_NoCollision )

def _disj(model, i, k, j):
    return [ model.NoCollision[i,k,j,ik] for ik in [0,1] ]
model.disj = Disjunction(model.L, rule=_disj)
```

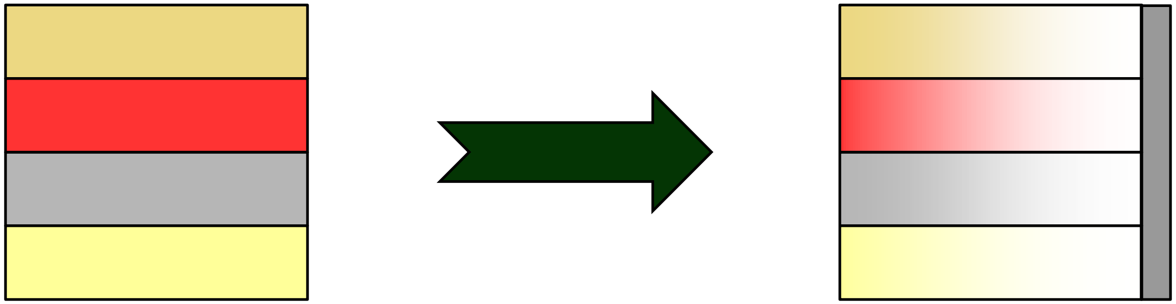
$$\left[\begin{array}{c} Y_{ik} \\ t_i + \sum_{\substack{m \in J(i) \\ m<j}} \tau_{im} + \tau_{ij} \le t_k + \sum_{\substack{m \in J(k) \\ m<j}} \tau_{km} \end{array} \right] \vee \left[\begin{array}{c} Y_{ki} \\ t_k + \sum_{\substack{m \in J(k) \\ m<j}} \tau_{km} + \tau_{kj} \le t_i + \sum_{\substack{m \in J(i) \\ m<j}} \tau_{im} \end{array} \right]$$

$$\forall j \in C_{ik}, \forall i,k \in I, i<k$$

Sandia National Laboratories

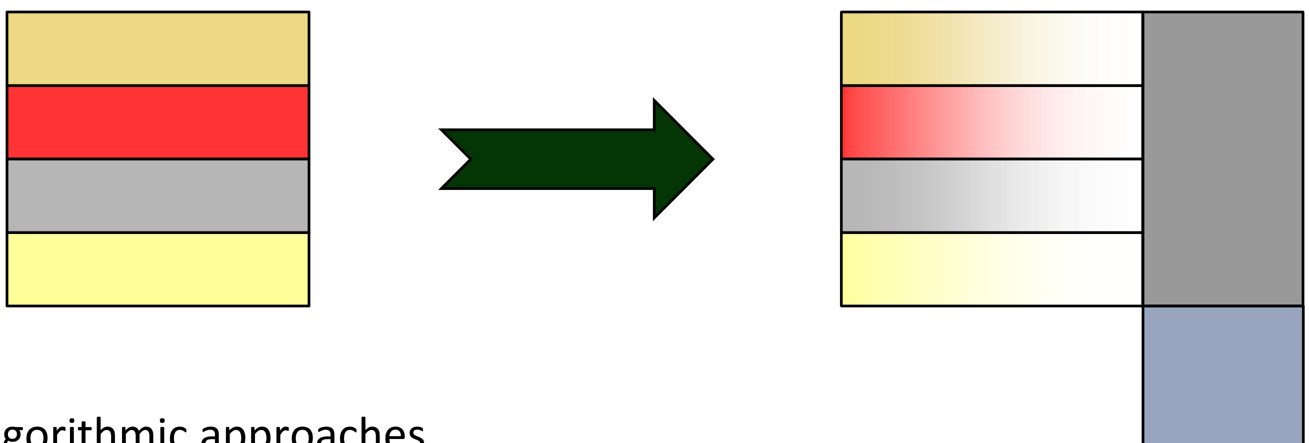
Solving disjunctive models

- Few solvers "understand" disjunctive models
  - *Transform* model into standard math program
  - Big-M relaxation:
    - Convert logic variables to binary
    - Split equality constraints in disjuncts into pairs of inequality constraints
    - Relax all constraints in the disjuncts with "appropriate" M values



Why is the transformation interesting?

- Model preserves explicit disjunctive structure
- Automated transformation reduces errors
- Automatically identifies appropriate M values (for bounded linear)
- Big-M is not the only way to relax a disjunction!
  - Convex hull transformation (Balas, 1985; Lee and Grossmann, 2000)



  - Algorithmic approaches
    - e.g., Trespalacios and Grossmann (submitted 2013)
  - Prematurely choosing one relaxation makes trying others difficult

Sandia National Laboratories

Expression transformations: MPEC

- Mathematical Programming with Equilibrium Constraints (MPEC)
  - Engineering design, economic equilibrium, multilevel games
  - Feasible region may be nonconvex and disconnected

- Equilibrium Constraints
  - Variational inequalities
  - Complementarity conditions
  - Optimality conditions (for bilevel problems)

MPEC formulations

- General MPEC models can be expressed as

$$
\begin{array}{lc}
\min_{x\in\mathbb{R}^n} & f(x) \\
\text{s.t.} & h(x) = 0 \\
& a_i \le w_i(x) \le b_i \perp v_i(x) \quad i = 1 \ldots m
\end{array}
$$

- The last set of constraints are generalized mixed complementarity conditions (Ferris, Fourer, and Gay, '06), which have the form

$$
\begin{array}{lll}
\text{either } w_i(x) = a_i & \text{and} & v_i(x) \ge 0 \\
\text{or } w_i(x) = b_i & \text{and} & v_i(x) \le 0 \\
\text{or } a_i < w_i(x) < b_i & \text{and} & v_i(x) = 0
\end{array}
$$

Modeling languages support MPECs

- AMPL
  - The `complements` keyword is used to denote complementarity between two constraints, expressions or variables
- GAMS
  - The `complements` keyword is used to denote complementarity between two constraints, expressions or variables
- AIMMS
  - Express mixed complementarity conditions by declaring complementarity variables along with associated constraints
- YALMIP
  - The `complements` function declares a constraint that reflects a mixed complementarity condition.

➢ Common challenge: lack of control over how the complementarity constraints are exposed to the solver

Sandia National Laboratories

Expressing complementarity conditions in Coopr

```
from coopr.pyomo import *
from coopr.mpec import Complementarity

M = ConcreteModel()
M.x = Var(bounds=(-1,2))
M.y = Var()

M.c3 = Complementarity(expr=(M.y - M.x**2 + 1 >= 0, M.y >= 0))
```

- The `Complementarity` component declares a complementarity condition
- The tuple argument specifies the two constraints, expressions, or variables in the complementarity condition.

This model definition is solver agnostic!

Sandia National Laboratories

A simple nonlinear reformulation

$$
\begin{array}{lll}
\min & f(x) & \\
s.t. & h(x) = 0 & \\
 & a_i \le \omega_i \le b_i & i = 1 \dots m \\
 & \omega_i = w_i(x) & i = 1 \dots m \\
 & (\omega_i - a_i) v_i(x) \le 0 & i = 1 \dots m \\
 & (\omega_i - b_i) v_i(x) \le 0 & i = 1 \dots m
\end{array}
$$

- NOTE: There are serious difficulties with solving this formulation as standard stability assumptions are not met.
  - But other nonlinear transformations exist!

Sandia National Laboratories

A simple disjunctive reformulation

$$
\begin{array}{lcr}
\min & f(x) & \\
\text{s.t.} & h(x) = 0 & \\
& \begin{bmatrix} y_{1,i} \\ w_i(x) = a_i \\ v_i(x) \geq 0 \end{bmatrix} \vee \begin{bmatrix} y_{2,i} \\ w_i(x) = b_i \\ v_i(x) \leq 0 \end{bmatrix} \vee \begin{bmatrix} y_{3,i} \\ a_i < w_i(x) < b_i \\ v_i(x) = 0 \end{bmatrix} & i = 1 \dots m \\
& y_{1,i} + y_{2,i} + y_{3,i} = 1 & i = 1 \dots m \\
& y_{1,i}, y_{2,i}, y_{3,i} \in \{0,1\} & i = 1 \dots m
\end{array}
$$

Back to our original example: ABS(x)

- Chaining transformations

$$
f = abs(x) \Rightarrow \begin{array}{c} f = x^+ + x^- \\ x = x^+ - x^- \\ x^+ \ge 0 \perp x^- \ge 0 \end{array} \Rightarrow \begin{array}{c} f = x^+ + x^- \\ x = x^+ - x^- \\ \begin{bmatrix} Y \\ x^- = 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y \\ x^+ = 0 \end{bmatrix} \\ x^+ \ge 0,\ x^- \ge 0 \end{array} \Rightarrow \begin{array}{c} f = x^+ + x^- \\ x = x^+ - x^- \\ x^- \le My \\ x^- \le M(1-y) \\ x^+ \ge 0,\ x^- \ge 0 \end{array}
$$

```
model = ConcreteModel()
# […]
TransformFactory("abs.complements").apply(model, inplace=True)
TransformFactory("mpec.disjunctive").apply(model, inplace=True)
TransformFactory("gdp.bigm").apply(model, inplace=True)
```


Sandia National Laboratories

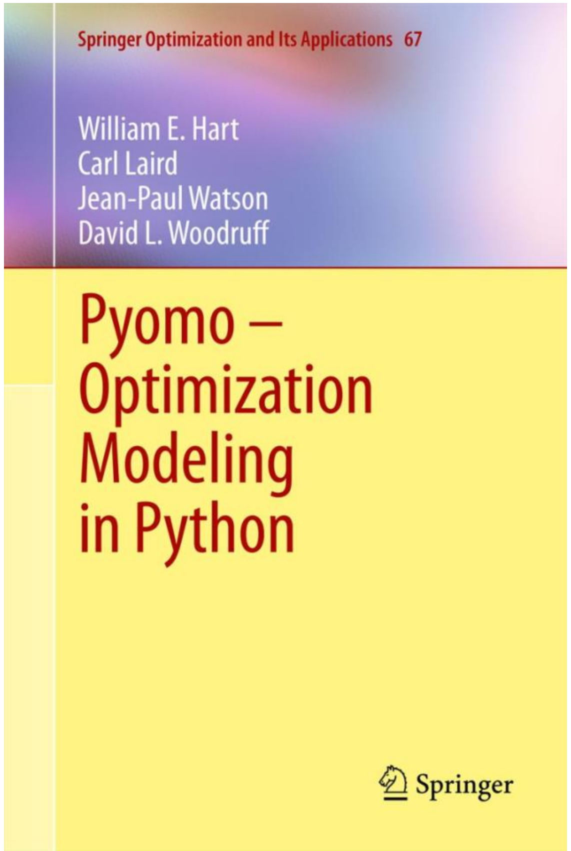
Summary

- Model transformations can significantly impact modeling
  - Separates the intent of the Modeler from the needs of the solver
  - Expands the set of (high-level) modeling constructs
    - Models can closer represent how a Modeler "thinks"
  - Defers decisions on how to map the problem class to the solver to just before solve time
  - Reduces / eliminates manual transcription errors
  - Chaining transformations is a powerful operation
    - Complex transformations are cast as a series of simpler operations
    - Availability of alternative transformation routes is preserved

- Other applications
  - Stochastic programming
  - Bilinear relaxations / linearizations
  - Bilevel model reformulation
  - DAE discretization

For more information…

- Project homepage
  - https://software.sandia.gov/coopr
- Mailing lists
  - "coopr-forum" Google Group
  - "coopr-developers" Google Group
- "The Book"



- Mathematical Programming Computation paper:
  - Pyomo: Modeling and Solving Mathematical Programs in Python (3(3), 2011)

Sandia National Laboratories