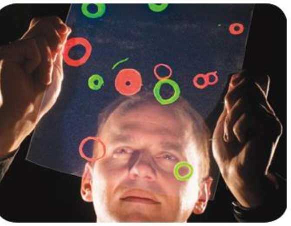
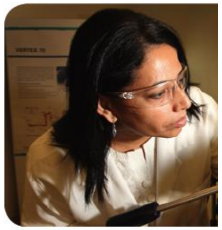




Sandia National Laboratories

Exceptional service in the national interest

Extended Math Programming as a framework for CPS models and analysis

John D. Siirola and Carl D. Laird

Discrete Math & Optimization (1464)
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM USA

CPS Ed Workshop, Paris, France
17 July 2017

U.S. DEPARTMENT OF ENERGY   NNSA National Nuclear Security Administration   CCR Center for Computing Research

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.

SAND2017-7362 C

Summary

- While not a “Grand Unifying Theory” model of CPS systems, Extended Math Programming is a useful paradigm for modeling and analyzing CPS systems
  - What do I mean by “Extended Math Programming”
    - Math programming
    - Analysis workflows
    - Model transformations
  - Extensions to Math Programming most relevant to CPS
    - Generalized Disjunctive Programming (GDP)
    - Dynamic systems (DAEs)
    - Stochastic programming
  - CPS Applications
    - Power grid operations and modeling
    - Computational approaches to Game Theory (for MTD)

What is optimization?

What is optimization?

- Finding the best answer!
  - "What is the lowest spot on the earth?"
    "**-39,944 ft**"

$$\min_x x$$

- Finding the *inputs* that give me the best answer!
  - "Where is the lowest spot on the earth?"
    "**Challenger Deep, Mariana Trench**"

$$\arg\min_x f(x)$$

- Finding the *valid inputs* that give me the best answer!
  - "Where is the lowest dry spot on earth?"
    "**The Dead Sea shoreline (-1391 ft)**"

$$\begin{aligned} &\arg\min_x f(x) \\ s.t.\quad & g(x) \le 0 \\ & h(x) = 0 \end{aligned}$$

What is optimization?

- Finding the best answer!
  - "What is the lowest spot on the earth?"
    "**-39,944 ft**"

$$\min_x x$$

- Finding the *inputs* that give me the best answer!
  - "Where is the lowest spot on the earth?"
    "**Challenger Deep, Mariana Trench**"

$$\arg\min_x f(x)$$

- Finding the *valid inputs* that give me the best answer!
  - "Where is the lowest dry spot on earth?"
    "**The Dead Sea shoreline (-1391 ft)**"

$$\begin{aligned} &\arg\min_x f(x) \\ s.t.\quad & g(x) \le 0 \\ & h(x) = 0 \end{aligned}$$

Constrained Optimization Models

- Wandering around in the real world, looking for the lowest spot is *expensive*, *time-consuming*, and *error-prone*
- We would rather work with a *model* of the real world
  - Represent what we know about the problem in a usable form
  - Incorporate assumptions and simplifications
  - Be both tractable and valid
    - (although these are often contradictory goals)
- *Mathematical Programming* is a convenient modeling paradigm:

$$\left.\begin{aligned} &\arg\min_x \; f(x) \\ &s.t. \quad g(a,x) \le 0 \\ &\qquad\;\; h(a,x) = 0 \end{aligned}\right\}$$

…but the details of *f*, *g*, and *h* dramatically impact the optimization algorithm!

- Supports *data agnostic modelling*

Model + Data → Instance

The universe of “math programming”

Operations Research

		If $f(x), g(a,x), h(a,x)$ are...	
		Linear	Nonlinear
If x is...	Continuous	**LP** [linear programming]	**NLP** [nonlinear programming]
	Discrete	**IP** [integer programming]	
	Continuous + Discrete	**MIP** [mixed integer (linear) programming]	**MINLP** [mixed integer nonlinear programming]

Physical systems
- dynamic systems
- process models
- uncertainty quantification

Systems design
Discrete operations
Cyber-physical systems

What's the problem with Math Programming?

Sandia National Laboratories

$$\text{Minimize}: \quad \sum_t \sum_g \left(c_g P_{g0t} + c_g^{SU} v_{gt} + c_g^{SD} w_{gt}\right)$$

$$\text{S.t.} \quad \theta^{\min} \le \theta_{nct} \le \theta^{\max}, \quad \forall\, n, c, t$$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{g0t} = d_{nt},$$

$$\forall\, n, \quad c = 0, \text{ transmission contingency states } c, t$$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{gct} = d_{nt},$$

$$\forall\, n, \text{ generator contingency states } c, t$$

$$P_{kc}^{\min} N1_{kc} z_{kt} \le P_{kct} \le P_{kc}^{\max} N1_{kc} z_{kt}, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} + (2 - z_{kt} - N1_{kc}) M_k \ge 0, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} - (2 - z_{kt} - N1_{kc}) M_k \le 0, \quad \forall\, k, c, t$$

$$P_g^{\min} N1_{gc} u_{gt} \le P_{gct} \le P_g^{\max} N1_{gc} u_{gt}, \quad \forall\, g, c, t$$

$$v_{g,t} - w_{g,t} = u_{g,t} - u_{g,t-1}, \quad \forall\, g, t$$

$$\sum_{q=t-UT_g+1}^{t} v_{g,q} \le u_{g,t}, \quad \forall\, g, t \in \{UT_g, \ldots, T\}$$

$$\sum_{q=t-DT_g+1}^{t} w_{g,q} \le 1 - u_{g,t}, \forall\, g, t \in \{DT_g, \ldots, T\}$$

$$P_{g0t} - P_{g0,t-1} \le R_g^{+} u_{g,t-1} + R_g^{SU} v_{g,t}, \quad \forall\, g, t$$

$$P_{g0,t-1} - P_{g0,t} \le R_g^{-} u_{g,t} + R_g^{SD} w_{g,t}, \quad \forall\, g, t$$

$$P_{gct} - P_{g0,t} \le R_g^{+}, \quad \forall\, g, c, t$$

$$P_{g0,t} N1_{gc} - P_{gct} \le R_g^{-}, \quad \forall\, g, c, t$$

$$0 \le v_{g,t} \le 1, \quad \forall\, g, t$$

$$0 \le w_{g,t} \le 1, \quad \forall\, g, t$$

$$u_{g,t} \in \{0, 1\}, \quad \forall\, g, t$$

- The MP "toolbox"
  - $+,\ -,\ \times,\ \div$
  - sin, cos, tan, *etc.*
  - $y^x,\ e^x,\ \log_{10}(x), \ln(x)$
  - *(functions in C^2)*

Hedman, et al., "Co-Optimization of Generation Unit Commitment and Transmission Switching With N-1 Reliability," *IEEE Trans Power Systems*, 25(2), pp.1052-1063, 2010

CCR Center for Computing Research

The previous slide is a real model…

- (In the US) Sequential markets (run by ISO/RTO):
  - “Unit commitment” (UC) / “Day-ahead Market” (DAM)
    - MIP run ~10 hours before the start of a day
    - Sets on/off state for all generator units hourly for 24 hours
  - “Reliability Unit Commitment” (RUC)
    - MIP run ~8 hours before the start of the day
    - Commits additional generators to meet spinning reserve and reliability (N-1 robustness) requirements
  - “Economic Dispatch” (ED) / “Security-constrained ED” (SCED)
    - “Real-time” markets: LP run hourly / every 5 minutes
    - Set generation levels, prices to meet realized demand

- Can switching lines on/off improve resilience / reduce cost?

- Problem scale
  - 100’s – 1000’s of buses; 2-3x lines

The Challenge: MP is dense and subtle

$$\text{Minimize}: \sum_t \sum_g \left(c_g P_{g0t} + c_g^{SU} v_{gt} + c_g^{SD} w_{gt}\right) \tag{1}$$

$$\text{S.t.} \quad \theta^{\min} \le \theta_{nct} \le \theta^{\max}, \quad \forall\, n, c, t \tag{2}$$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{g0t} = d_{nt},$$
$$\forall\, n, \quad c = 0, \text{ transmission contingency states } c, t \tag{3a}$$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{gct} = d_{nt},$$
$$\forall\, n, \text{ generator contingency states } c, t \tag{3b}$$

$$P_{kc}^{\min} N1_{kc} z_{kt} \le P_{kct} \le P_{kc}^{\max} N1_{kc} z_{kt}, \quad \forall\, k, c, t \tag{4}$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} + (2 - z_{kt} - N1_{kc}) M_k \ge 0, \quad \forall\, k, c, t \tag{5a}$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} - (2 - z_{kt} - N1_{kc}) M_k \le 0, \quad \forall\, k, c, t \tag{5b}$$

$$P_g^{\min} N1_{gc} u_{gt} \le P_{gct} \le P_g^{\max} N1_{gc} u_{gt}, \quad \forall\, g, c, t \tag{6}$$

$$v_{g,t} - w_{g,t} = u_{g,t} - u_{g,t-1}, \quad \forall\, g, t \tag{7}$$

$$\sum_{q=t-UT_g+1}^{t} v_{g,q} \le u_{g,t}, \quad \forall\, g, t \in \{UT_g, \ldots, T\} \tag{8}$$

$$\sum_{q=t-DT_g+1}^{t} w_{g,q} \le 1 - u_{g,t}, \forall\, g, t \in \{DT_g, \ldots, T\} \tag{9}$$

$$P_{g0t} - P_{g0,t-1} \le R_g^{+} u_{g,t-1} + R_g^{SU} v_{g,t}, \quad \forall\, g, t \tag{10}$$

$$P_{g0,t-1} - P_{g0,t} \le R_g^{-} u_{g,t} + R_g^{SD} w_{g,t}, \quad \forall\, g, t \tag{11}$$

$$P_{gct} - P_{g0,t} \le R_g^{+}, \quad \forall\, g, c, t \tag{12}$$

$$P_{g0,t} N1_{gc} - P_{gct} \le R_g^{-}, \quad \forall\, g, c, t \tag{13}$$

$$0 \le v_{g,t} \le 1, \quad \forall\, g, t \tag{14}$$

$$0 \le w_{g,t} \le 1, \quad \forall\, g, t \tag{15}$$

$$u_{g,t} \in \{0, 1\}, \quad \forall\, g, t \tag{16}$$

Hedman, et al., "Co-Optimization of Generation Unit Commitment and Transmission Switching With N-1 Reliability," *IEEE Trans Power Systems*, 25(2), pp.1052-1063, 2010

The Challenge: MP is dense and subtle

$$\text{Minimize}: \sum_t \sum_g \left(c_g P_{g0t} + c_g^{SU} v_{gt} + c_g^{SD} w_{gt}\right)$$

S.t.

$$\theta^{\min} \le \theta_{nct} \le \theta^{\max}, \quad \forall\, n, c, t$$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{g0t} = d_{nt},$$

$\forall\, n, \quad c = 0,$ transmission contingency states c, t

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{gct} = d_{nt},$$

$\forall\, n,$ generator contingency states c, t

$$P_{kc}^{\min} N1_{kc} z_{kt} \le P_{kct} \le P_{kc}^{\max} N1_{kc} z_{kt}, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} + (2 - z_{kt} - N1_{kc}) M_k \ge 0, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} - (2 - z_{kt} - N1_{kc}) M_k \le 0, \quad \forall\, k, c, t$$

$$P_g^{\min} N1_{gc} u_{gt} \le P_{gct} \le P_g^{\max} N1_{gc} u_{gt}, \quad \forall\, g, c, t$$

$$v_{g,t} - w_{g,t} = u_{g,t} - u_{g,t-1}, \quad \forall\, g, t$$

$$\sum_{q=t-UT_g+1}^{t} v_{g,q} \le u_{g,t}, \quad \forall\, g, t \in \{UT_g, \dots, T\}$$

$$\sum_{q=t-DT_g+1}^{t} w_{g,q} \le 1 - u_{g,t}, \forall\, g, t \in \{DT_g, \dots, T\}$$

$$P_{g0t} - P_{g0,t-1} \le R_g^{+} u_{g,t-1} + R_g^{SU} v_{g,t}, \quad \forall\, g, t$$

$$P_{g0,t-1} - P_{g0,t} \le R_g^{-} u_{g,t} + R_g^{SD} w_{g,t}, \quad \forall\, g, t$$

$$P_{gct} - P_{g0,t} \le R_g^{+}, \quad \forall\, g, c, t$$

$$P_{g0,t} N1_{gc} - P_{gct} \le R_g^{-}, \quad \forall\, g, c, t$$

$$0 \le v_{g,t} \le 1, \quad \forall\, g, t$$

$$0 \le w_{g,t} \le 1, \quad \forall\, g, t$$

$$u_{g,t} \in \{0, 1\}, \quad \forall\, g, t$$

To a first approximation:

- DCOPF
- Economic dispatch
- Unit commitment
- Transmission switching
- N-1 contingency

Hedman, et al., "Co-Optimization of Generation Unit Commitment and Transmission Switching With N-1 Reliability," *IEEE Trans Power Systems*, 25(2), pp.1052-1063, 2010

Sidebar: What do these have in common?

$$a = \sqrt{(x-3)^2 + \epsilon}$$

$$\begin{gathered} a = b + c \\ b \le M \cdot y \\ c \le M(1-y) \\ x - 3 = c - b \\ b \ge 0 \\ c \ge 0 \\ y \in \{0,1\} \end{gathered}$$

$$\begin{gathered} a \ge x - 3 \\ a \ge 3 - x \end{gathered}$$

$$a = \frac{2(x-3)}{1 + e^{-\frac{x-3}{h}}} - x + 3$$

$$\begin{gathered} a = b + c \\ x - 3 = c - b \\ b \ge 0 \perp c \ge 0 \end{gathered}$$

Sidebar: What do these have in common?

$$a = b + c$$
$$b \leq M \cdot y$$
$$c \leq M(1 - y)$$
$$x - 3 = c - b$$
$$b \geq 0$$
$$c \geq 0$$
$$y \in \{0,1\}$$

$$a = \sqrt{(x-3)^2 + \epsilon}$$

$$a = abs(x - 3)$$

$$a \geq x - 3$$
$$a \geq 3 - x$$

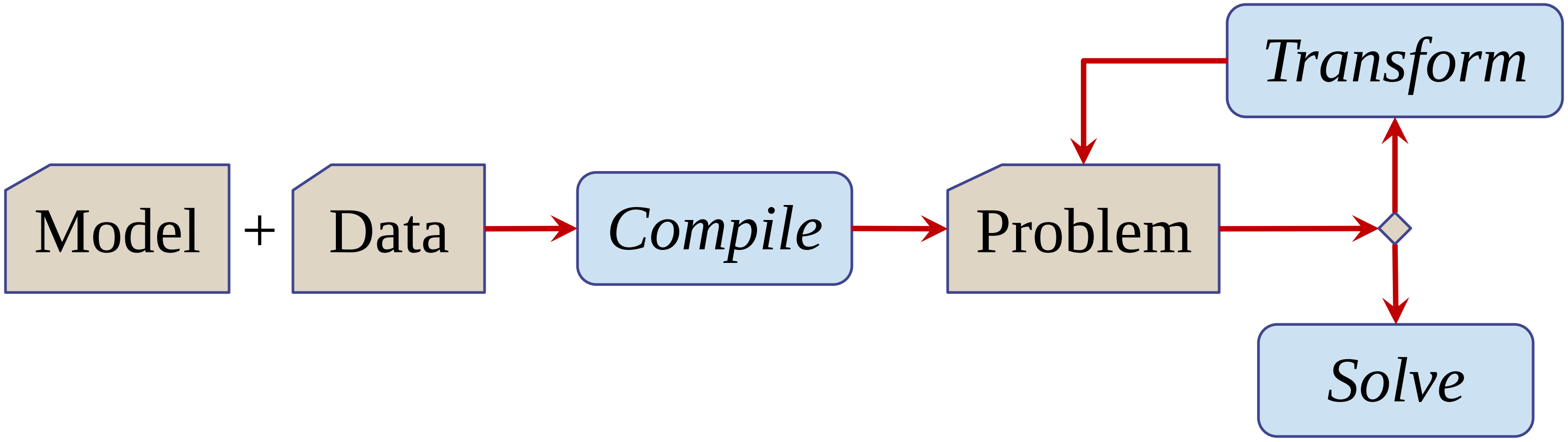
$$a = b + c$$
$$x - 3 = c - b$$
$$b \geq 0 \perp c \geq 0$$

$$a = \frac{2(x-3)}{1 + e^{-\frac{x-3}{h}}} - x + 3$$

If we *mean* "$a = abs(x - 3)$",
why don't we *write* that in our models???

A new solution workflow

- Model Transformations: *Projecting problems to problems*
  - Project from one problem space to another
  - Standardize common reformulations or approximations
  - Enables "Extended Math Programming"[1]
    - Develop new modeling constructs not supported by solvers
    - (Automatically) Convert these "unoptimizable" modeling constructs into equivalent optimizable forms



[1] - Ferris, et al. "An extended mathematical programming framework". *Computers & Chemical Engineering* 33(12) 2009.

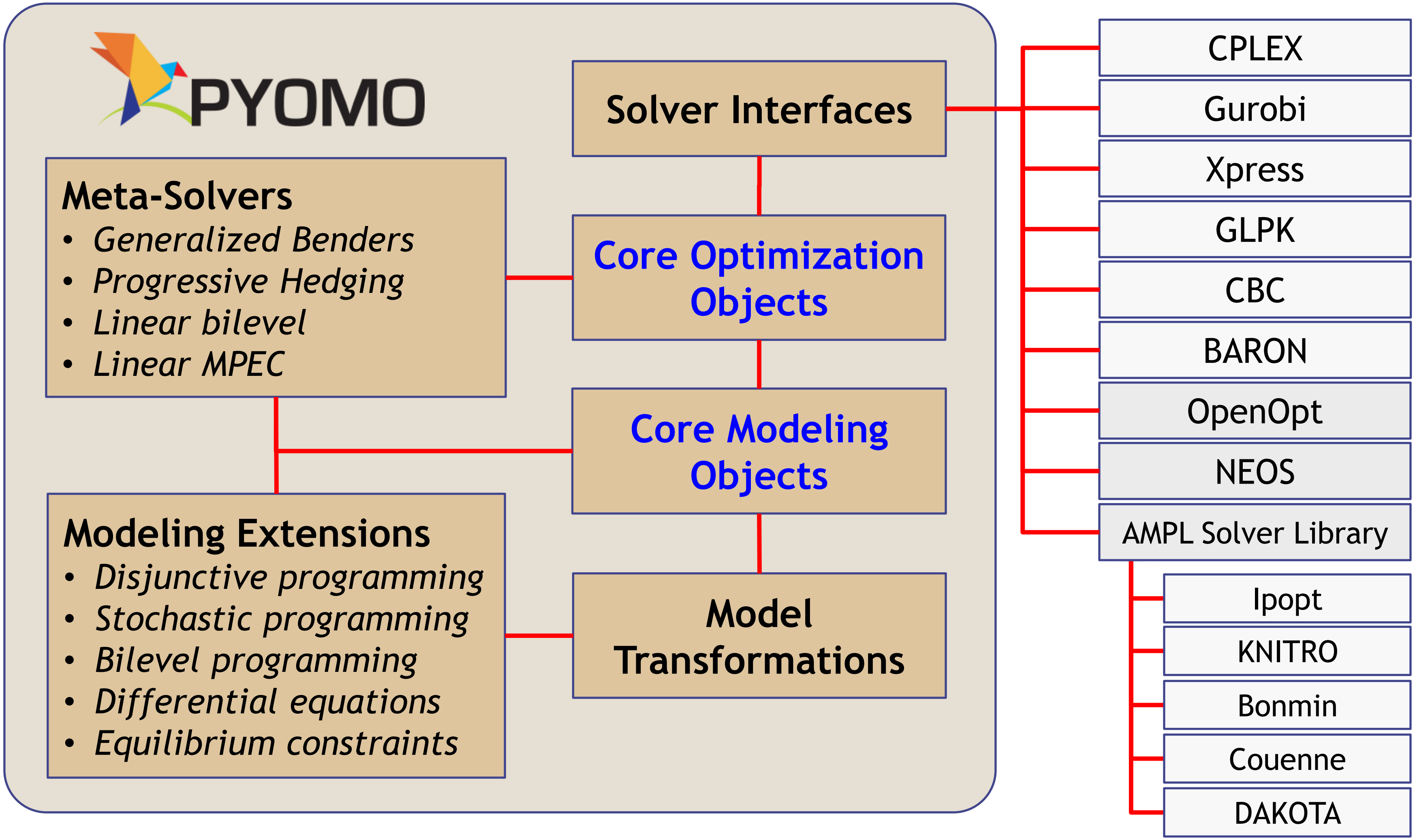
Extended Math Programming

- Ferris, et al. (2009)
  - Modeling framework (domain-specific language) built on GAMS
  - Adds *support for "higher level"* constructs
    - Complementarity conditions, Variational inequalities, Bilevel problems, Disjunctive programming
  - Constructs are annotated through a separate input file
  - Interfaces to specialized solvers or provides *automated reformulations* for standard solvers

- Alternatively, EMP concepts could be implemented through an *object-oriented* framework

PYOMO

Pyomo: optimization modeling in Python



Extended Math Programming for CPS

- Key (CPS) modeling needs
  - Modularity and composability
  - Continuous and discrete (logic-based) models
  - Continuous dynamics (physical systems)
  - Stochastic models / uncertainty quantification

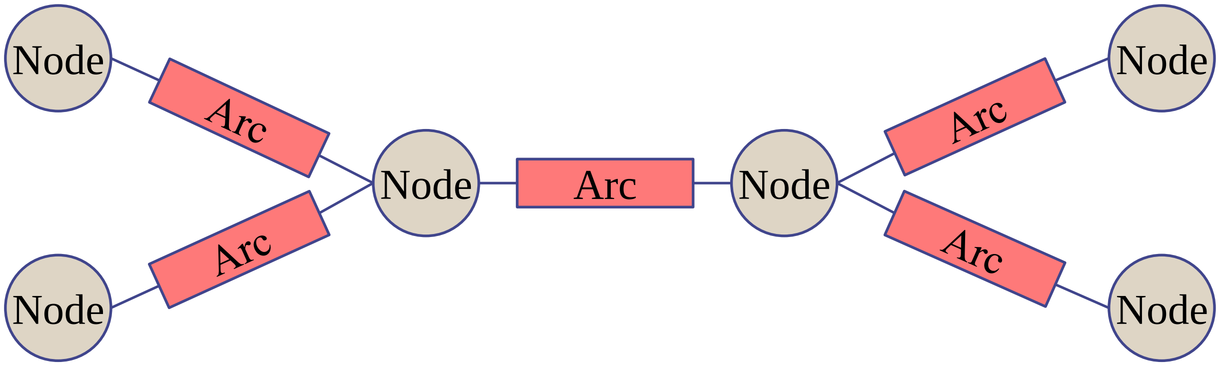
- EMP capabilities
  - Hierarchical model definitions
  - Complementarity conditions
  - Generalized Disjunctive Programming (GDP)
  - (Discretized) systems of differential-algebraic equations (DAE)
  - Stochastic programming / design under uncertainty

Block-oriented modeling

- "Blocks"
  - Collections of model components
    - Variable, Parameter, Set, Constraint, etc.
  - Blocks may be arbitrarily nested
- Why blocks?
  - Support reusable modeling components
  - Express distinctly modeled concepts as distinct objects
  - Manipulate modeled components as distinct entities
  - Explicitly expose model structure (e.g., for decomposition)
  - Enables transformations and component namespaces
- Prior art
  - Ubiquitous in the simulation community
  - Rare in Math Programming environments
    - *Notable exceptions:* ASCEND, JModelica.org

Structured modeling with blocks

- Capture connected block structure, e.g., *network flow*



- Blocks interface through *connectors* (group of variables)
- Block implementation independent of network definition

Domain	Node	Arc	Connector Vars
Fluid flow	Mass balance	Pressure Drop	Pressure; Volumetric flow
AC Power flow	KCL	Active power transfer; Reactive power transfer	Phase angle; Active power flow; Reactive power flow

Generalized disjunctive programming

- 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
      - (Variable, Parameter, Constraint, etc.)
    - Boolean (binary) indicator variable determines if block is enforced
  - `Disjunction`:
    - Enforces logical OR/XOR across a set of Disjunct indicator variables
  - Logic constraints on indicator variables

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

$$\Omega(Y) = true$$

Simple Example: Task sequencing in Pyomo

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

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

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

Simple Example: Task sequencing in Pyomo

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

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

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

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
      - Automatically calculate M values for linear expressions

```
pyomo solve --solver=cbc --transform=gdp.bigm jobshop.py jobshop.dat
```

  - Convex hull relaxation (Balas, 1985; Lee and Grossmann, 2000)
    - Disaggregate variables in all disjuncts
    - Bound disaggregated variables with Big-M terms

```
pyomo solve --solver=cbc --transform=gdp.chull jobshop.py jobshop.dat
```

CCR
Center for Computing Research

A transformation-centric view of *abs()*

- Chaining transformations

$$f = abs(x) \Rightarrow \begin{gathered} f = x^+ + x^- \\ x = x^+ - x^- \\ x^+ \geq 0 \perp x^- \geq 0 \end{gathered} \Rightarrow \begin{gathered} f = x^+ + x^- \\ x = x^+ - x^- \\ \begin{bmatrix} Y \\ x^- = 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y \\ x^+ = 0 \end{bmatrix} \\ x^+ \geq 0,\ x^- \geq 0 \end{gathered} \Rightarrow \begin{gathered} f = x^+ + x^- \\ x = x^+ - x^- \\ x^- \leq My \\ x^- \leq M(1-y) \\ x^+ \geq 0,\ x^- \geq 0 \end{gathered}$$

```
model = ConcreteModel()
# […]
TransformFactory(“abs.complements”).apply_to(model)
TransformFactory(“mpec.disjunctive”).apply_to(model)
TransformFactory(“gdp.bigm”).apply_to(model)
```

Extensions to *dynamic systems*

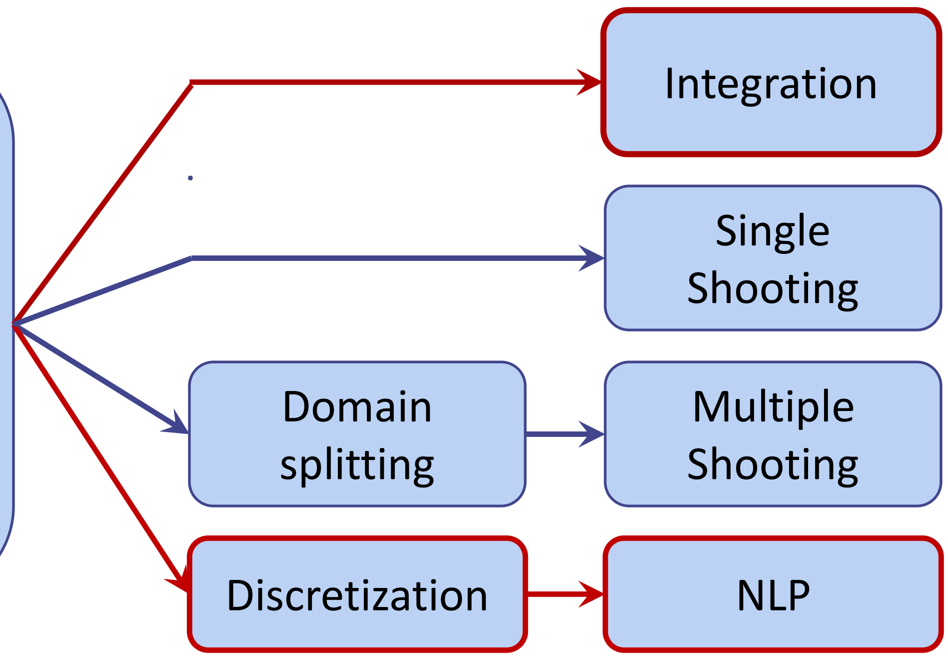
- Optimization of dynamic systems is *hard*.
  - In OR, think "multi-stage" problems
  - In "engineered systems", think differential equations
    - High fidelity *simulation* is difficult and expensive (e.g., HPC)
    - How to optimize?
      - Simulation-based optimization (*single shooting*)
      - Multiple shooting methods
      - Discretization (*collocation methods*)
    - Common theme: significant effort to rework formulation
      - Time: first ~6 months of a grad student's research
      - Error prone: many ways to make subtle mistakes
      - Inflexible: formulation specific to selected solution approach

Dynamic systems through EMP

- Model dynamical systems in a natural form
  - Systems of Differential Algebraic Equations (DAE)
  - Extend the Pyomo component model
    - `ContinuousSet`: A virtual set over which you can take a derivative
    - `DerivativeVar`: The derivative of a Var with respect to a ContinuousSet

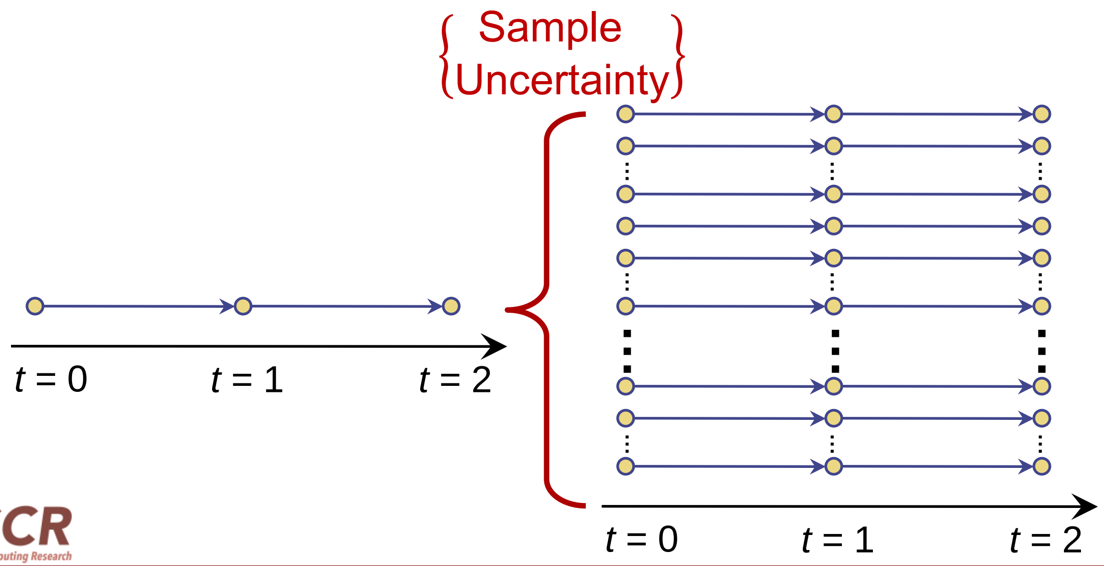
DAE model

$$\min \Psi(x(t), y(t), u(t))$$
$$\dot{x}(t) = f(x(t), y(t), u(t), t, p)$$
$$0 = g(x(t), y(t), u(t), t, p)$$
$$x(t) \in \Re^n$$
$$y(t) \in \Re^n$$
$$u(t) \in \Re^m$$



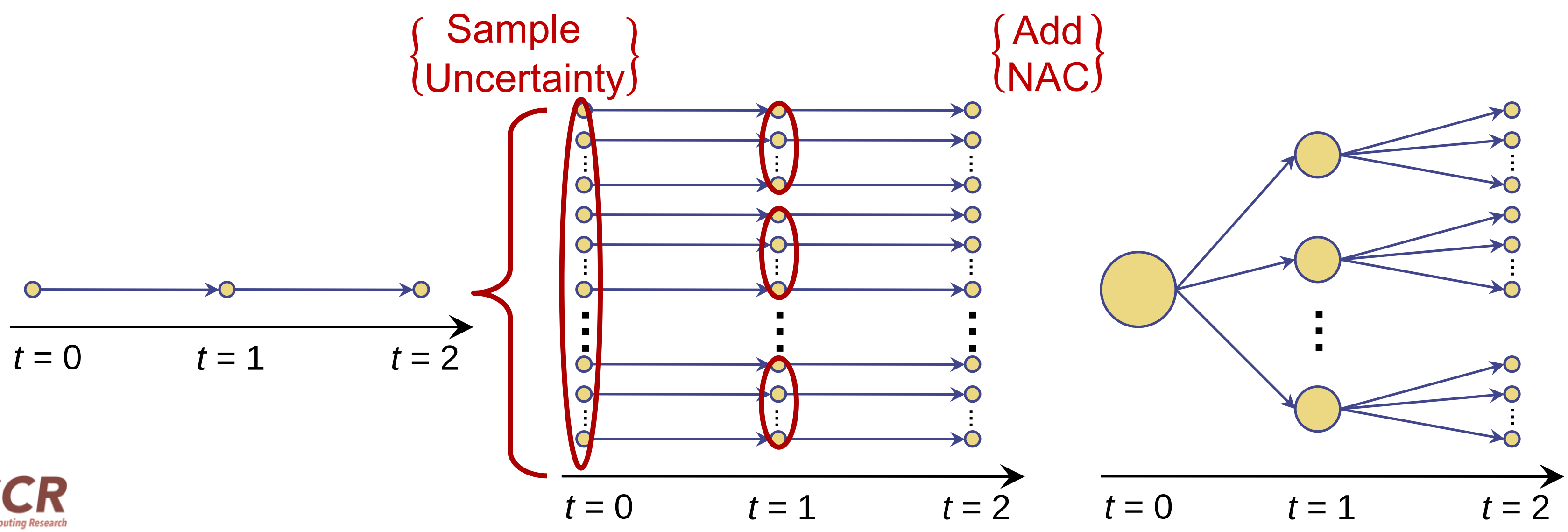
Optimization under uncertainty

- We see increasing demand for optimization under uncertainty
  - Recognition that decisions must explicitly incorporate risk
  - Many approaches: surrogates, sampling, robust optimization
  - *We focus on stochastic programming*
    - Capture problem uncertainty as a set of possible scenarios
    - Solve to select a single answer that optimizes across all scenarios
    - Naturally leverages a transformation-based approach



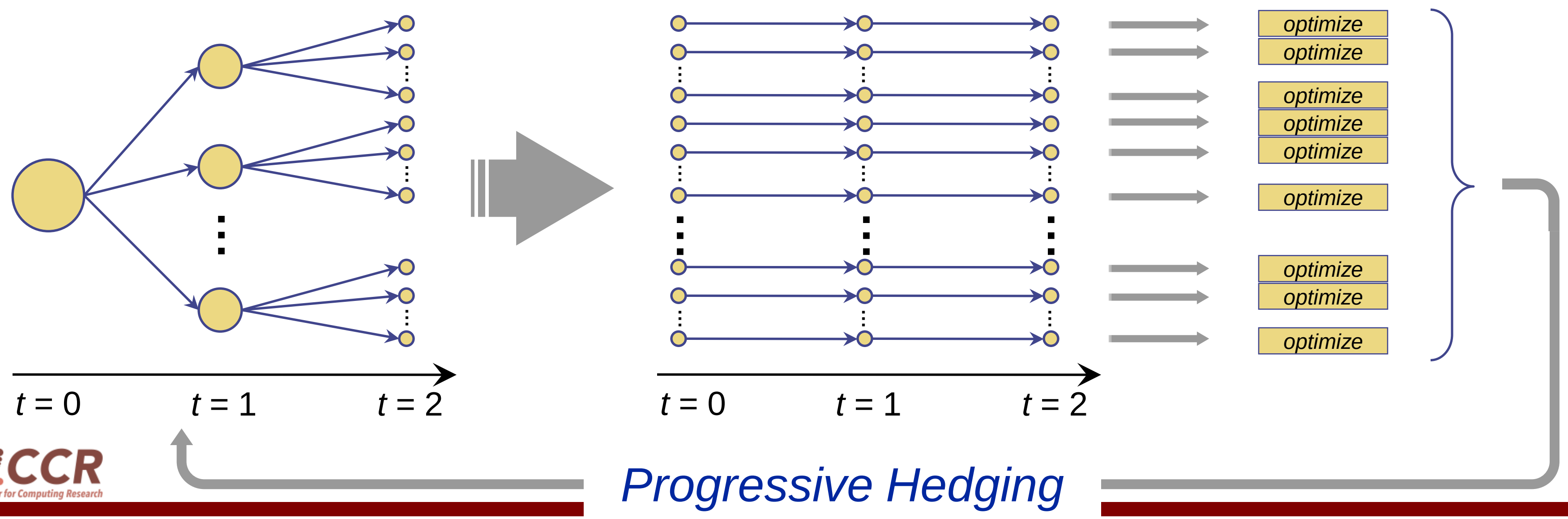
Optimization under uncertainty

- We see increasing demand for optimization under uncertainty
  - Recognition that decisions must explicitly incorporate risk
  - Many approaches: surrogates, sampling, robust optimization
  - *We focus on stochastic programming*
    - Capture problem uncertainty as a set of possible scenarios
    - Solve to select a single answer that optimizes across all scenarios
    - Naturally leverages a transformation-based approach



What if the problem is too difficult?

- Implement *meta algotithms* (via problem decomposition)!
  - Stage-wise (e.g., Benders decomposition [Benders, 1962])
    - Master problem (1st stage), independent (2nd stage) subproblems
    - Master problem grows with cuts from subproblems
  - Scenario-based (e.g., Progressive Hedging [Rockafellar & Wets, 1991])
    - "No" master problem
    - Iteratively converge NAC by penalizing deviation from consensus



Selected applications

- Reliability Unit Commitment with Transmission Switching
  - Enhance the resiliency of the electric transmission system by ensuring the system can survive the loss of any single asset (generator or non-radial transmission line)
- Evaluate cyber-motivated game theoretic models
  - Compute optimal defender strategies for notional adversarial models

Returning to RUC + Transmission Switching

Minimize : $\sum_t \sum_g (c_g P_{g0t} + c_g^{SU} v_{gt} + c_g^{SD} w_{gt})$

S.t. $\theta^{\min} \le \theta_{nct} \le \theta^{\max}, \quad \forall\, n, c, t$

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{g0t} = d_{nt},$$

$\forall\, n, \quad c = 0,$ transmission contingency states c, t

$$\sum_{\forall\, k(n,.)} P_{kct} - \sum_{\forall\, k(.,n)} P_{kct} + \sum_{\forall\, g(n)} P_{gct} = d_{nt},$$

$\forall\, n,$ generator contingency states c, t

$$P_{kc}^{\min} N1_{kc} z_{kt} \le P_{kct} \le P_{kc}^{\max} N1_{kc} z_{kt}, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} + (2 - z_{kt} - N1_{kc}) M_k \ge 0, \quad \forall\, k, c, t$$

$$B_k(\theta_{nct} - \theta_{mct}) - P_{kct} - (2 - z_{kt} - N1_{kc}) M_k \le 0, \quad \forall\, k, c, t$$

$$P_g^{\min} N1_{gc} u_{gt} \le P_{gct} \le P_g^{\max} N1_{gc} u_{gt}, \quad \forall\, g, c, t$$

$$v_{g,t} - w_{g,t} = u_{g,t} - u_{g,t-1}, \quad \forall\, g, t$$

$$\sum_{q=t-UT_g+1}^{t} v_{g,q} \le u_{g,t}, \quad \forall\, g, t \in \{UT_g, \ldots, T\}$$

$$\sum_{q=t-DT_g+1}^{t} w_{g,q} \le 1 - u_{g,t}, \forall\, g, t \in \{DT_g, \ldots, T\}$$

$$P_{g0t} - P_{g0,t-1} \le R_g^{+} u_{g,t-1} + R_g^{SU} v_{g,t}, \quad \forall\, g, t$$

$$P_{g0,t-1} - P_{g0,t} \le R_g^{-} u_{g,t} + R_g^{SD} w_{g,t}, \quad \forall\, g, t$$

$$P_{gct} - P_{g0,t} \le R_g^{+}, \quad \forall\, g, c, t$$

$$P_{g0,t} N1_{gc} - P_{gct} \le R_g^{-}, \quad \forall\, g, c, t$$

$$0 \le v_{g,t} \le 1, \quad \forall\, g, t$$

$$0 \le w_{g,t} \le 1, \quad \forall\, g, t$$

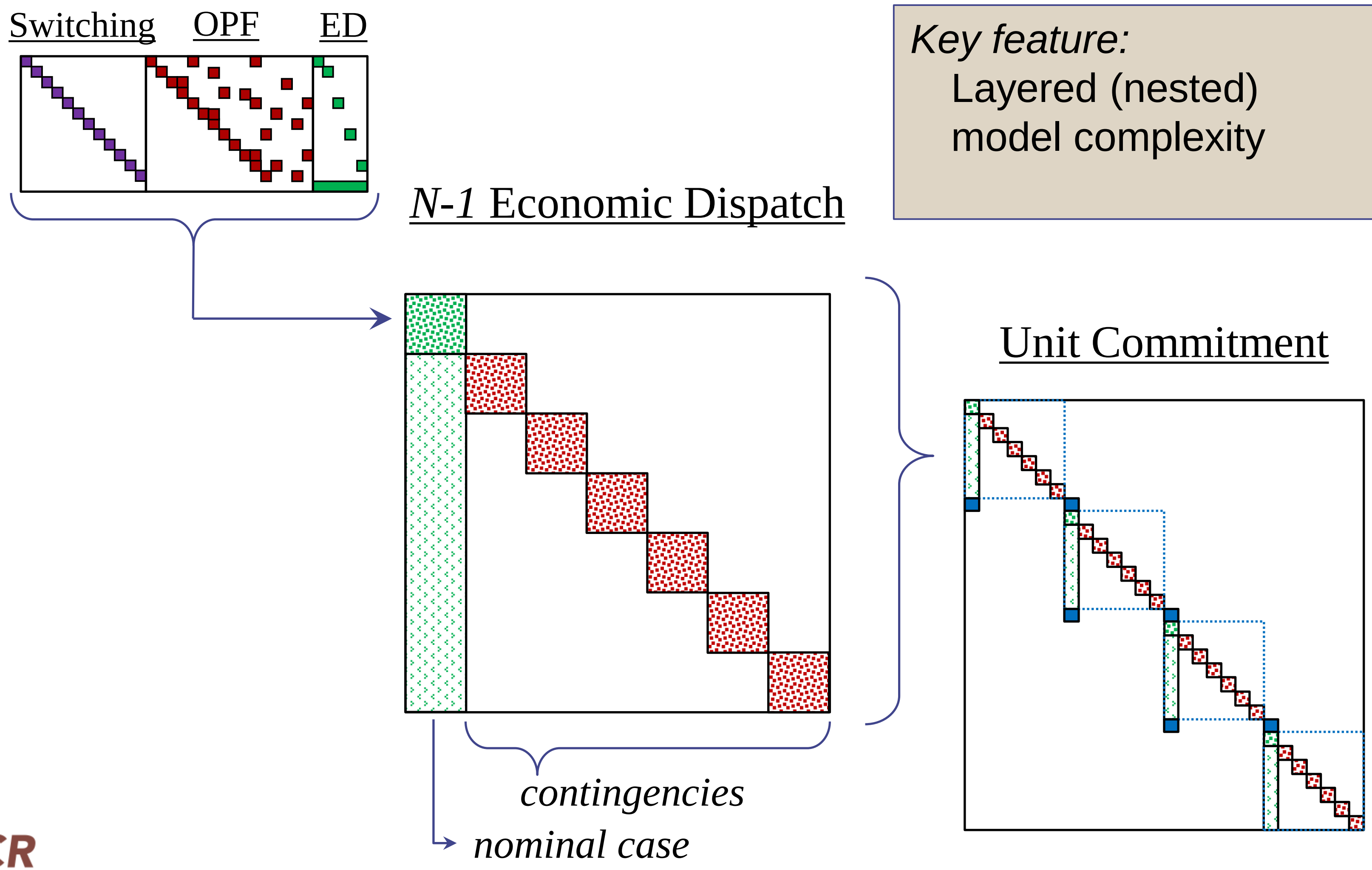
$$u_{g,t} \in \{0, 1\}, \quad \forall\, g, t$$

To a first approximation:

- DCOPF
- Economic dispatch
- Unit commitment
- Transmission switching
- N-1 contingency

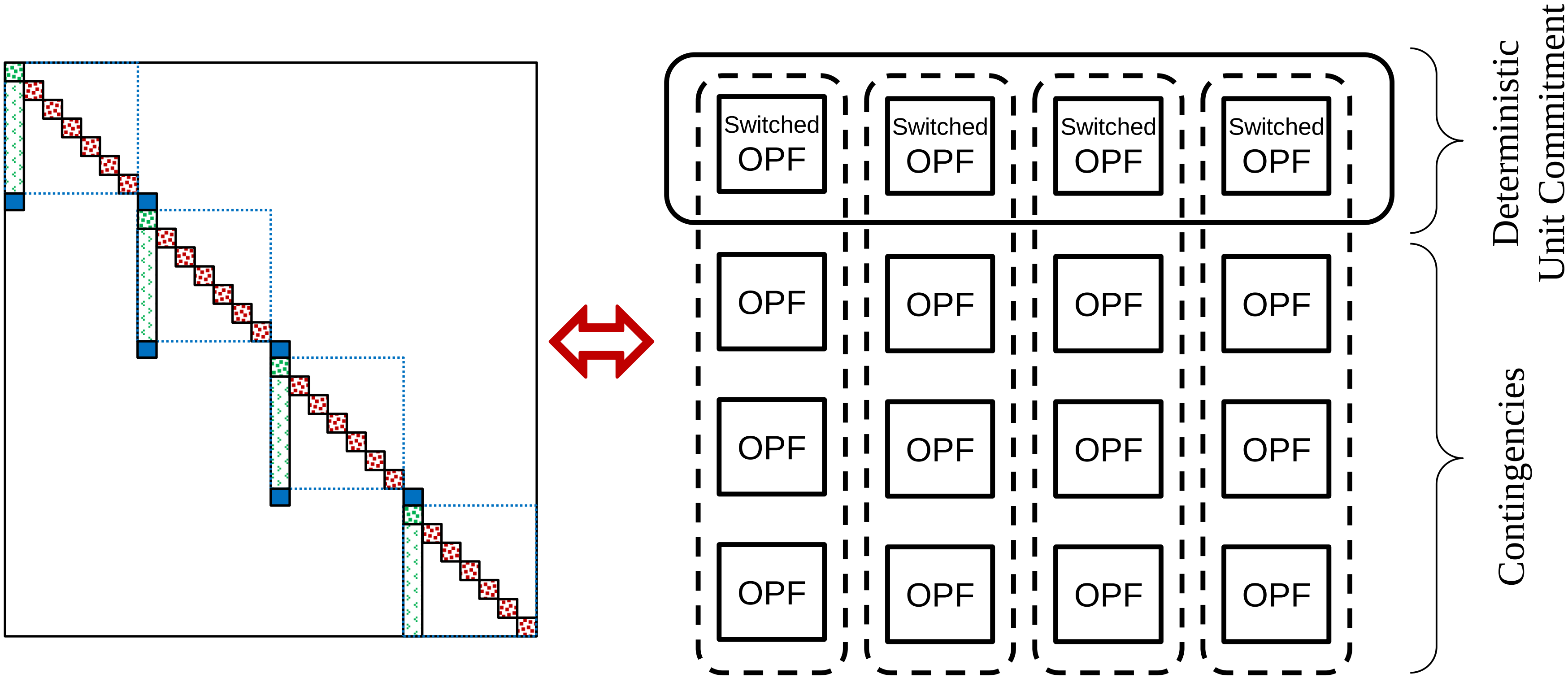
Hedman, et al., "Co-Optimization of Generation Unit Commitment and Transmission Switching With N-1 Reliability," *IEEE Trans Power Systems*, 25(2), pp.1052-1063, 2010

(Nonobvious) Inherent structure



Key feature:
Layered (nested) model complexity

This still doesn't *quite* tell the whole story



Explicitly expose disjunctive decisions

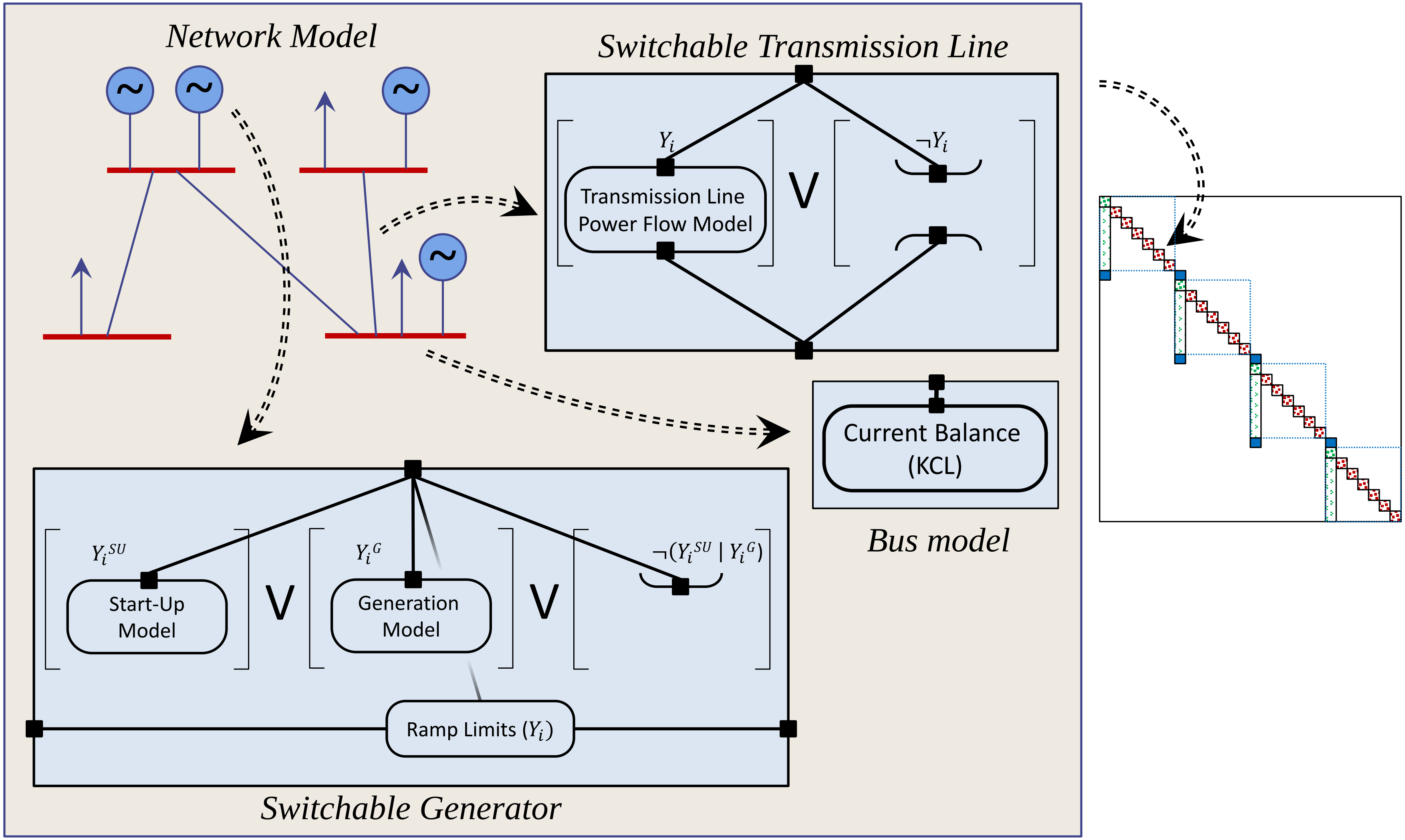
- Transmission switching:

$$\begin{bmatrix} z_{kct} \\ P_{kct} = B_k(\theta_{k1} - \theta_{k2}) \end{bmatrix} \vee \begin{bmatrix} \neg z_{kct} \\ P_{kct} = 0 \end{bmatrix}$$

- Generation

$$\begin{bmatrix} u_{gt} \\ C_{gt} = P_{gt} c_g \\ R_g^{+} \geq P_{gt} - P_{gt-1} \\ R_g^{-} \geq P_{gt-1} - P_{gt} \end{bmatrix} \vee \begin{bmatrix} v_{kt} \\ C_{gt} = P_{gt} c_g + c_g^{SU} \\ R_g^{SU} \geq P_{gt} - P_{gt-1} \end{bmatrix} \vee \begin{bmatrix} \neg(u_{kt} \mid v_{kt}) \\ C_{gt} = c_g^{SD} u_{kt-1} \\ R_g^{SD} \geq P_{gt-1} - P_{gt} \\ P_{gt} = 0 \end{bmatrix}$$

Embed within a structured model



Optimal Solution of RTS-96

- From Hedman, et al. 2010
  - N-1 UC solution: 3,245,997
  - N-1 UC w/ Switching: 3,125,185 (2 pass UC+switching heuristic)

	Rows	Columns	Binaries
Raw model	5,118,760	1,501,177	5,184
After presolve	2,634,851	1,062,290	4,476

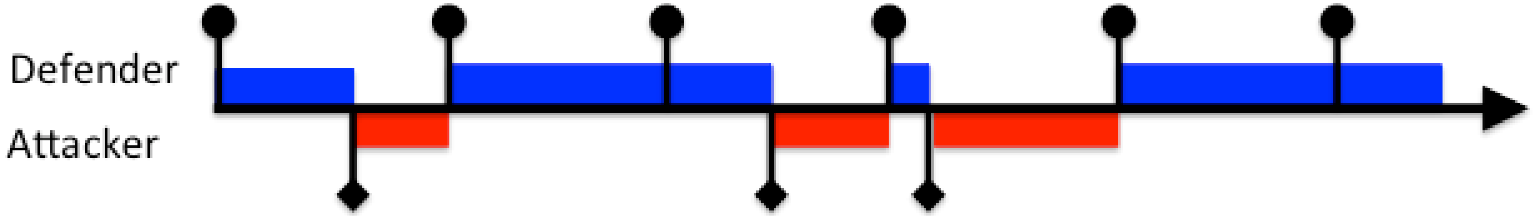
- Restructured problem (complete N-1 UC w/ switching):

	Rows	Columns	Binaries
Raw model	21,232,224	13,129,692	3,796,830
After presolve	2,471,714	1,249,976	187,194

  - Solution (1e-4 gap): 2,990,004 (60,000 sec)
  - Automated Big-M relaxation (including automatic M calculation)
  - Default solver settings

Modeling Attacker-Defender Games

- Capture high-level aspects of real system defense
- Simplest example: FlipIt, “stealthy takeover”
  - Two players: attacker and defender
  - One contested resource. Defender holds at start
  - A player can move at a cost
    - Takes resource (tie to defender)
    - Neither player ever knows who owns the resource
  - Strategy: when to move? Timeline is infinite.
  - Utility = (time in control) – cost (can be weighted)
  - Many results in the original paper



Marten van Dijk, Ari Juels, and Alina Oprea. FlipIt: The Game of “Stealthy Takeover”. Journal of Cryptology, 26(4):655–713, 2013.

Exploring Alternatives to *Simulation –vs– Analytical*

- Analysis continuum

Simulation	Stochastic Programming	Analytical
Increasing Flexibility, Expressiveness		Increasing Generality

- Challenges:
  - Analytical: optimal response over continuous (infinite) parameters
    - May require restrictive / unrealistic assumptions (e.g., periodic moves)
  - Simulation: enumerate (subset of) parameters and collect statistics
    - Search by full enumeration frequently computationally intractable
- Opportunity:
  - Leverage numerical optimization to gain prescriptive insights while preserving much of the flexibility of simulation

Stochastic Programming

- Key idea in stochastic programming:
  - approximate uncertainty by sampling outcomes
- Approximate attacker's strategy space by sampling possible random success-time outcomes
  - Attack scenarios
  - More scenarios gives a better approximation
- Optimize to determine the defender's single best strategy against ALL scenarios
  - Non-anticipative (only one solution for all attacks)
- Extensive form is a mixed-integer program (MIP)
- Can express more easily as a disjunctive program (DP)
  - Convert DP to MIP

Cases

For each scenario s and time t, only 3 possible cases:

- Attacker takes over (defender doesn't move)

$$a_{st} = 1 \text{ and } d_t = 0 \text{ and } \rho_{st} = 0$$

- Defender takes over

$$d_t = 1 \text{ and } \rho_{st} = 1$$

- Nothing changes

$$a_{st} = 0 \text{ and } d_t = 0 \text{ and } \rho_{st} = \rho_{s,t-1}$$

- Where
  - a_{st}: Attacker moves at time t in attack scenario s
  - d_t: Defender moves at time t
  - $\rho_{st} = \begin{cases} 1 \text{ if defender controls resource at time } t \\ 0 \text{ if attacker controls resource at time } t \end{cases}$

FlipIt Disjunctive Program

$$
\begin{aligned}
\max \quad & |T|^{-1}|S|^{-1} \sum_{s\in S, t\in T} \rho_{s,t} - c_{take}\sum_{t\in T} d_t \\
s.t. \quad & \begin{bmatrix} Y_{1,s,t} \\ a_{s,t} = 0 \\ d_t = 0 \\ \rho_{s,t} = \rho_{s,t-1} \end{bmatrix} \bigvee \begin{bmatrix} Y_{2,s,t} \\ a_{s,t} = 1 \\ d_t = 0 \\ \rho_{s,t} = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_{3,s,t} \\ \\ d_t = 1 \\ \rho_{s,t} = 1 \end{bmatrix} \\
& & \forall s \in S, \{t | t \in T, t > 0\} \\
& Y_{1,s,t} + Y_{2,s,t} + Y_{3,s,t} = 1 & \forall s \in S, \{t | t \in T, t > 0\} \\
& \rho_{s,0} = 1 & \forall s \in S \\
& d_t \in \{0,1\} & \forall t \in T \\
& \rho_{s,t} \in \{0,1\} & \forall s \in S, t \in T \\
& Y_{i,s,t} \in \{0,1\} & \forall i \in \{1,2,3\}, s \in S, t \in T
\end{aligned}
$$

- Pick a case for each scenario at each time
- The Pyomo modeling language lets you write it about this way

Equivalent MIP

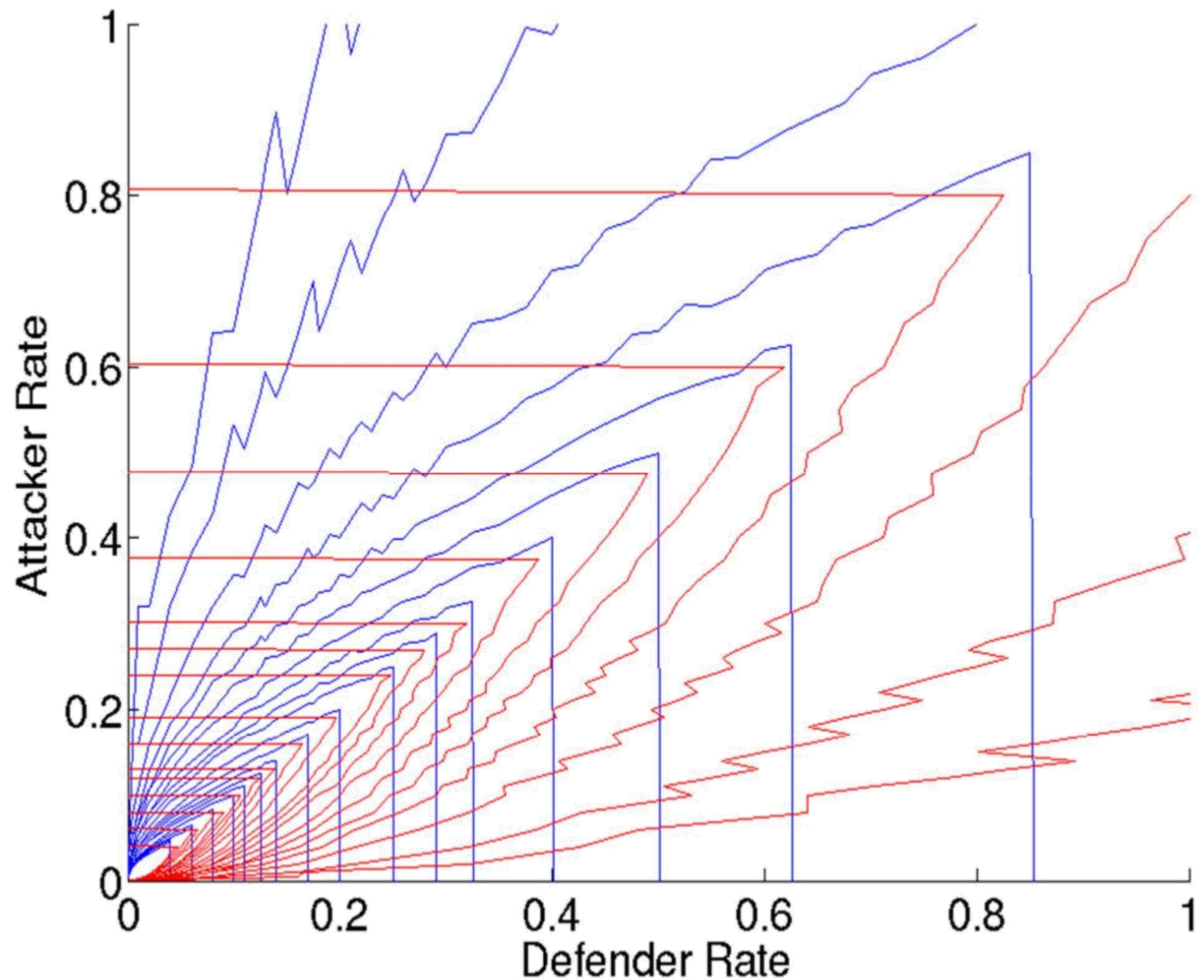
- Pyomo can (automatically) translate to a model form like this

$$
\text{(FlipIt)} \quad \text{minimize} \quad \sum_{t \in T, s \in S} (\rho_{st} - c_{\text{move}} d_t)
$$

$$
\text{where} \quad \begin{cases}
y_{st}^{(a)} \leq a_{st} & \forall t \in T, s \in S \\
y_{st}^{(a)} + y_{st}^{(d)} \geq a_{st} & \forall t \in T, s \in S \\
d_t \leq \sum_{s \in S} y_{st}^{(a)} & \forall t \in T \\
\rho_{st} \leq 1 - y_{st}^{(a)} & \forall t \in T, s \in S \\
y_{st}^{(d)} = d_t & \forall t \in T, s \in S \\
\rho_{st} \geq y_{st}^{(d)} & \forall t \in T, s \in S \\
a_{st} - d_t \leq 1 - y_{st}^{(z)} & \forall t \in T, s \in S \\
a_{st} - d_t \geq y_{st}^{(z)} - 1 & \forall t \in T, s \in S \\
\rho s0 = 1 & \forall s \in S \\
\rho st - \rho s, t - 1 \leq 1 - y_{st}^{(z)} & \forall t \in T, s \in S \\
\rho st - \rho s, t - 1 \geq y_{st}^{(z)} - 1 & \forall t \in T, s \in S \\
y_{st}^{(z)} + y_{st}^{(a)} + y_{st}^{(d)} = 1 & \forall t \in T
\end{cases}
$$

Families of Best-Response Curves

- Even a modest time horizon (64) and number of scenarios (32) approximates infinite game



- Varying defender move cost and benefit of holding resource.
- Attacker Blue
- Defender Red

Summary

- While not a “Grand Unifying Theory” model of CPS systems, Extended Math Programming is a useful paradigm for modeling and analyzing CPS systems
  - What do I mean by “Extended Math Programming”
    - Math programming
    - Analysis workflows
    - Model transformations
  - Extensions to Math Programming most relevant to CPS
    - Generalized Disjunctive Programming (GDP)
    - Dynamic systems (DAEs)
    - Stochastic programming
    - Bilevel optimization
  - CPS Applications
    - Power grid operations and modeling
    - Computational approaches to Game Theory (for MTD)

EMP is *not* a panacea

- Extended Math Programming is a useful framework for consistently expressing CPS; *however*,
  - The ability to express the problem does not guarantee a solution
    - e.g., a minor extension to FlipIt yields a game (PLADD) that is resistant to direct solution by stochastic programming
  - Problems can scale beyond abilities of current solvers
    - LP: > 1e8; MIP: > 1e7; NLP: > 1e6; MINLP > 1e3?
    - But algorithms are advancing
      - At least as fast as computing [Amundson 1988, Bixby 2012]
      - Decomposition, formulation engineering, specialized solvers
- The formalism, expressiveness, and rigor has pedagogical value.
  - *Could* form the basis of a GUT for classroom settings
  - *Extensible* to new CPS-specific modeling constructs
  - *Rich algorithm research space*

Thank you!

- For more information…
- Project homepages
  - http://www.pyomo.org
  - http://software.sandia.gov/pyomo
- User mailing lists
  - pyomo-forum@googlegroups.com
- “The Book”
  - Second Edition now available!
- Mathematical Programming Computation papers
  - Pyomo: Modeling and Solving Mathematical Programs in Python (Vol. 3, No. 3, 2011)
  - PySP: Modeling and Solving Stochastic Programs in Python (Vol. 4, No. 2, 2012)