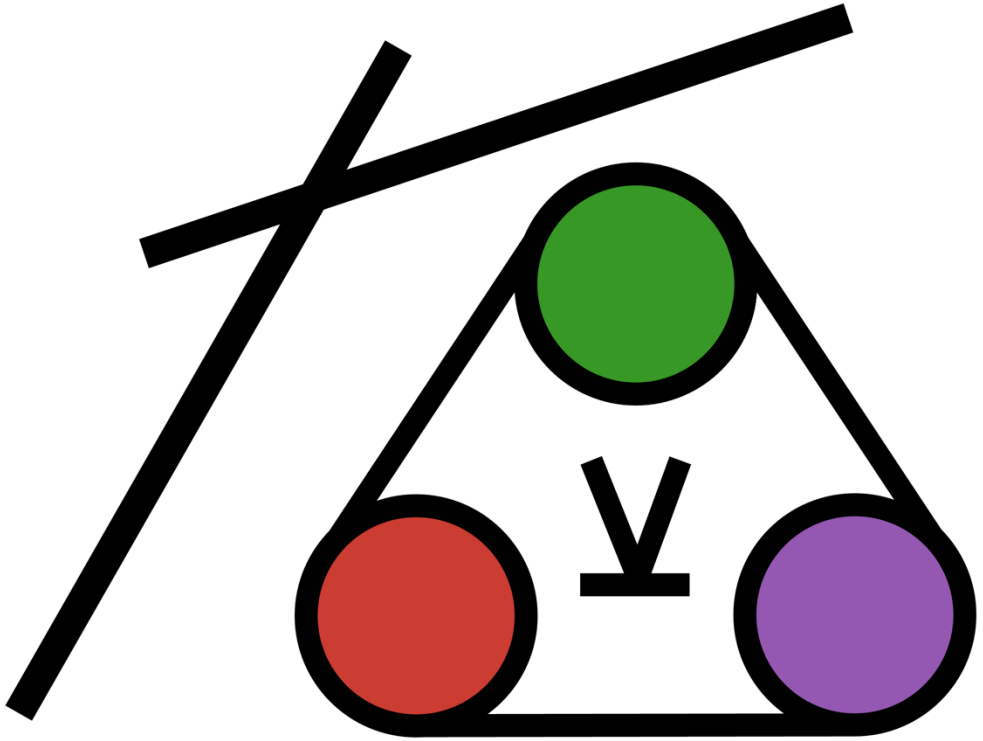
DISJUNCTIVEPROGRAMMING.JL'S NEW SUITE OF METHODS

01/06/2026



Daniel Nguyen & Joshua Pulsipher,
Department of Chemical Engineering

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Contributors

Hector Perez, PhD

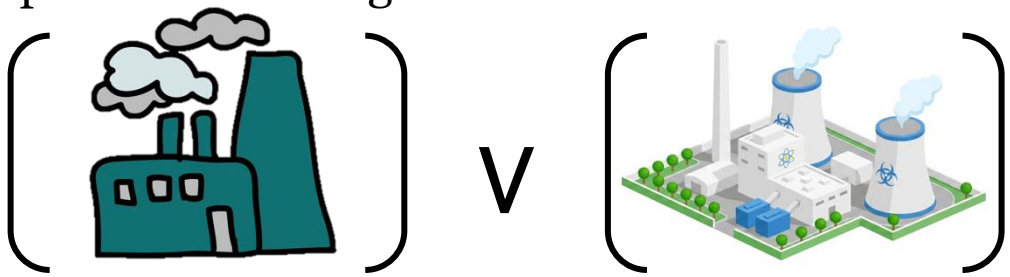
Professor Joshua Pulsipher

Evelyn Gondosiswanto, MASc

DISJUNCTIVE PROGRAMMING OVERVIEW

Logic Constraints

- Example: Expansion Planning



- Propositional Logic:

AND	OR	NOT	IF	ONLY IF
$\wedge$	$\vee$	$\neg$	$\Rightarrow$	$\Leftrightarrow$

- Constraint Programming Logic:

  atleast$(n, \cdot)$ exactly$(n, \cdot)$ atmost $(n, \cdot)$

- Represented via **Disjuncts** activated by **Boolean Variables**

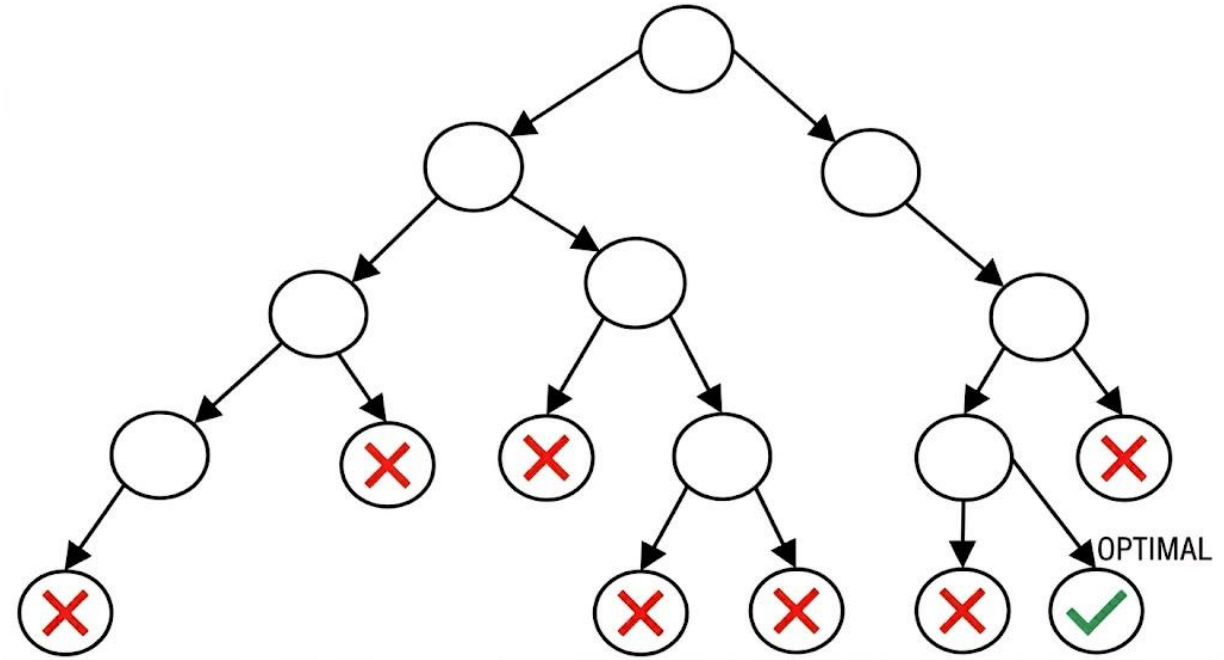
- **Conditionally enforce constraints based on values of Boolean vars!**

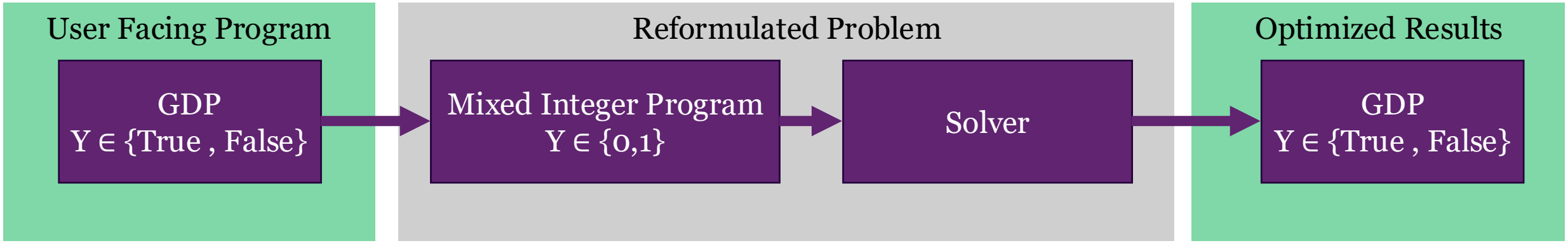
Generalized Disjunctive Program

$$
\begin{aligned}
\min_{x,Y} \quad & f(x) \\
\text{s.t.} \quad & g(x) \leqslant 0 \\
& \bigvee_{j \in J_i} \begin{bmatrix} Y_{ij} \\ r_{ij}(x) \leqslant 0 \end{bmatrix}, \quad i \in \mathcal{I} \\
& \Omega(Y) = \text{True} \\
& x \in \mathcal{X}, \; Y_{ij} \in \{\text{True}, \text{False}\}
\end{aligned}
$$

How to solve a GDP

- Solvers **CANNOT** solve GDP directly
- Solvers take Mixed Integer Programs (MIPs)
- Use methods like branch and bound
  - Integrality subject to relaxation
- GDPs have MIP counterparts
- Recover an interpretable GDP





REFORMULATION TECHNIQUES

GDP Form

$$\min_{\mathbf{x},\,Y} \quad f(x_1, x_2)$$

$$\text{s.t.} \quad \begin{bmatrix} Y_1 \\ (x_1-1)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\vee$$

$$\begin{bmatrix} Y_2 \\ (x_1-4)^2 + (x_2-4.5)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\vee$$

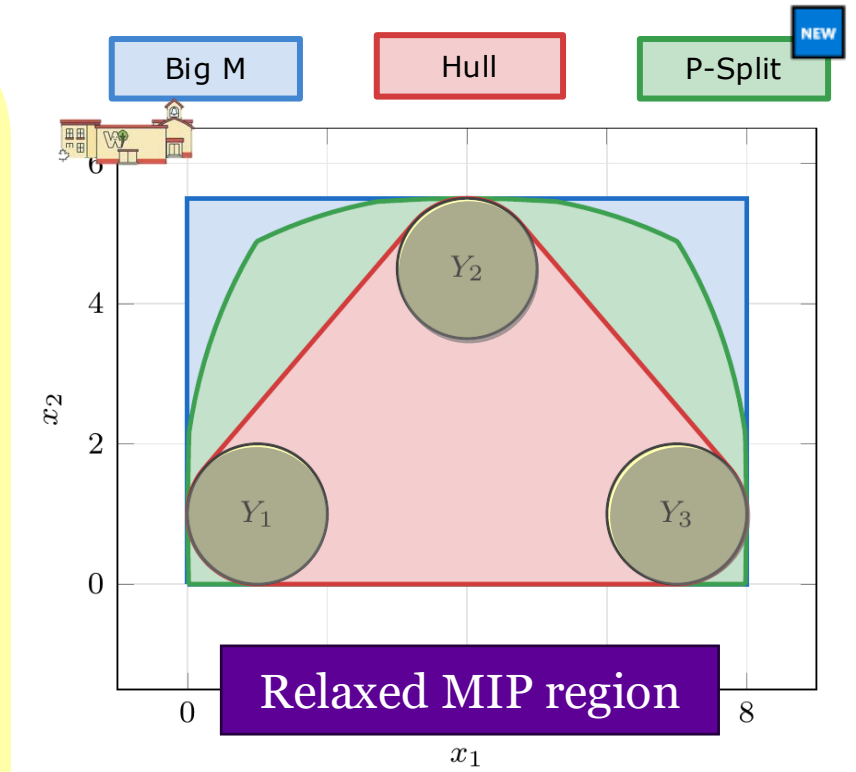
$$\begin{bmatrix} Y_3 \\ (x_1-7)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\text{exactly}(1, Y)$$

$$0 \leqslant x_1 \leqslant 8, \quad 0 \leqslant x_2 \leqslant 5.5$$

$$-1 \leqslant x_3 \leqslant 1, \quad -1 \leqslant x_4 \leqslant 1$$

$$Y_i \in \{\text{True, False}\}$$



MIP Form (Big M)

$$\min \quad f(x_1, x_2)$$

$$\text{s.t.} \quad (x_1-1)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 + M_1(1-Y_1)$$

$$(x_1-4)^2 + (x_2-4.5)^2 + x_3^2 + x_4^2 \leqslant 1 + M_2(1-Y_2)$$

$$(x_1-7)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 + M_3(1-Y_3)$$

$$0 \leqslant x_1 \leqslant 8, \quad 0 \leqslant x_2 \leqslant 5.5$$

$$-1 \leqslant x_3 \leqslant 1, \quad -1 \leqslant x_4 \leqslant 1$$

$$Y_1 + Y_2 + Y_3 = 1, \quad Y_i \in \{0, 1\}$$

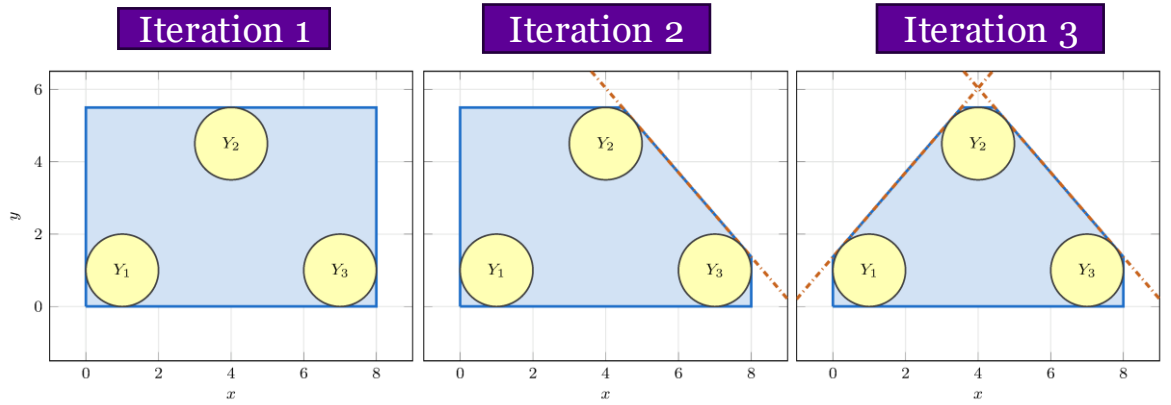
REFORMULATION TECHNIQUES

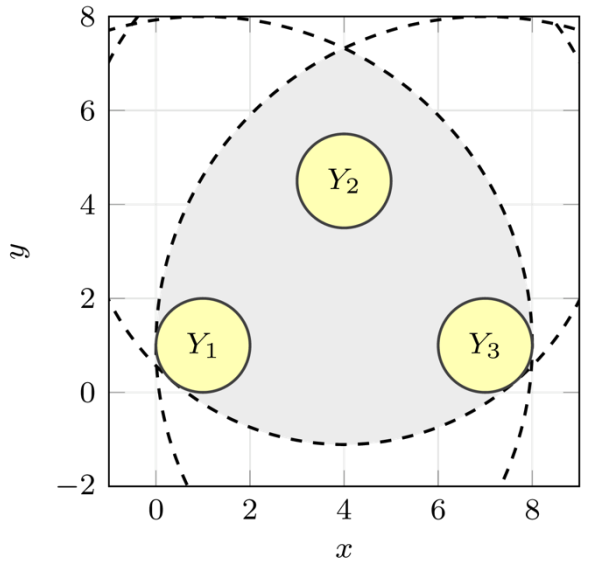
GDP ➔ MIP ➔ Solve

- Some methods require **subproblems**

GDP ➔ **Subproblems** ➔ MIP ➔ Solve

- **Cutting Planes** solves for linear cuts NEW
- **Multiple Big M** solves for coefficients NEW





Big-M **(3 constants total: M_1, M_2, M_3)**

$$r_1(z) \leqslant M_1\,(1 - y_1)$$
$$r_2(z) \leqslant M_2\,(1 - y_2)$$
$$r_3(z) \leqslant M_3\,(1 - y_3)$$

MBM **(6 constants total: $M_{12}, M_{13}, M_{21}, M_{23}, M_{31}, M_{32}$)**

$$r_1(z) \leqslant M_{12}\,y_2 + M_{13}\,y_3$$
$$r_2(z) \leqslant M_{21}\,y_1 + M_{23}\,y_3$$
$$r_3(z) \leqslant M_{31}\,y_1 + M_{32}\,y_2$$

Each M_{ij} is optimized for

DisjunctiveProgramming.jl

GDP Form

$$\min_{x, Y} \quad f(x_1, x_2)$$

s.t.

$$\begin{bmatrix} Y_1 \\ (x_1-1)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\vee$$

$$\begin{bmatrix} Y_2 \\ (x_1-4)^2 + (x_2-4.5)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\vee$$

$$\begin{bmatrix} Y_3 \\ (x_1-7)^2 + (x_2-1)^2 + x_3^2 + x_4^2 \leqslant 1 \end{bmatrix}$$

$$\text{exactly}(1, Y)$$

$$0 \leqslant x_1 \leqslant 8, \quad 0 \leqslant x_2 \leqslant 5.5$$
$$-1 \leqslant x_3 \leqslant 1, \quad -1 \leqslant x_4 \leqslant 1$$

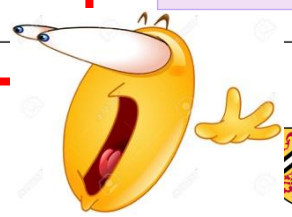
$$Y_i \in \{\text{True, False}\}$$

```julia
using DisjunctiveProgramming, Gurobi

model = GDPModel(Gurobi.Optimizer)

lb = [ 0.0,  0.0, -1.0, -1.0]
ub = [ 8.0,  5.5,  1.0,  1.0]

@variable(model, lb[k] <= x[k = 1:4] <= ub[k])

I = 1:3
@variable(model, Y[I], Logical)

c = [(1.0, 1.0, 0.0, 0.0),
     (4.0, 4.5, 0.0, 0.0),
     (7.0, 1.0, 0.0, 0.0)]

f(x1, x2) = (x1 - 0.0)^2 + (x2 - 5.0)^2

@constraint(model, [i = I],
    sum((x[k] - c[i][k])^2 for k = 1:4) <= 1,
    Disjunct(Y[i]))

@disjunction(model, Y)
@objective(model, Min, f(x[1], x[2]))

optimize!(model, gdp_method = Hull())
```

Supported Reformulations:

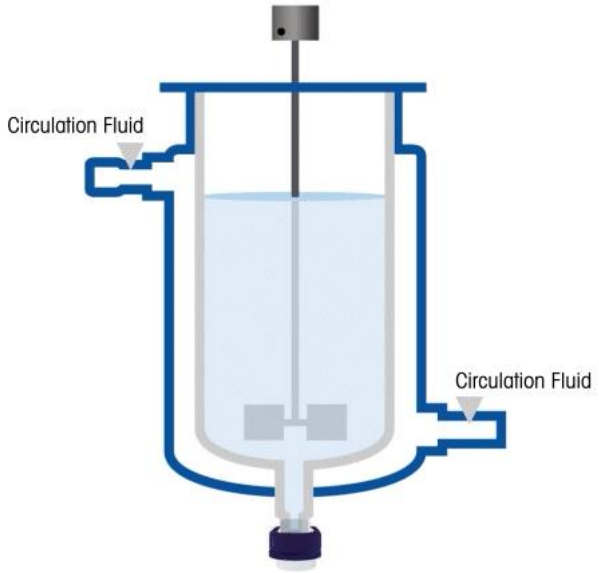
Big-M
Hull
Indicator
P-Split NEW
Cutting Planes NEW
Multiple Big-M NEW

and their Infinite versions!



What is Infinite-dimensional Optimization (Infiniteopt)?

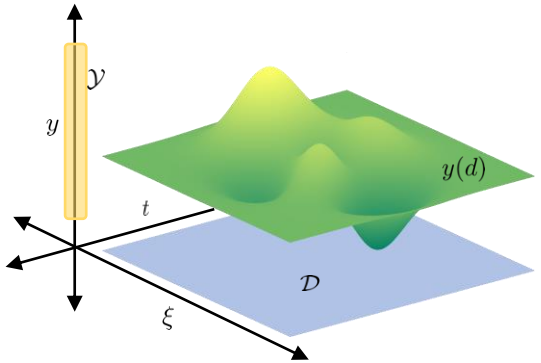
- Many real-world processes evolve over **continuous domains** like time and space
  - Chemical processes, rocket trajectory, portfolio optimization, etc…
  - Involves decision variables that are **infinite-dimensional functions**
  - **Example:** Optimal control of a jacketed reactor
- Implemented in `InfiniteOpt.jl`



Decision variables:

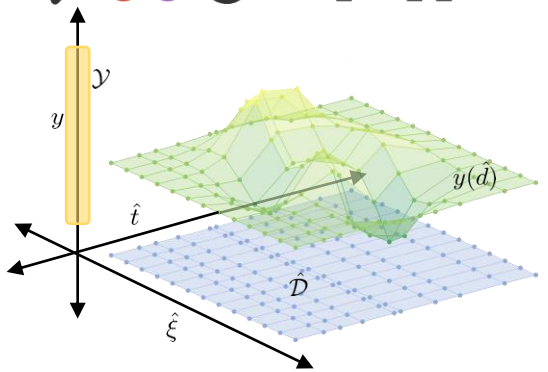
$C_A(t)$, $T_{tank}(t)$, $T_{jacket}(t)$

InfiniteOpt



Direct Transcription

JUMP



UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A *Unifying Modeling Abstraction for Infinite-Dimensional Optimization*." Computers & Chem. Eng. 2022
Mettler Toledo. *"Jacketed Reactors."* 2025. URL: https://www.mt.com/us/en/home/products/L1_AutochemProducts/automated-reactor-control-systems/jacketed-reactors.html

INFINITE DISJUNCTIVE PROGRAMS

- Modeling GDPs over infinite domains

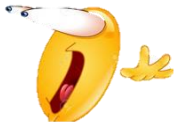
$$\bigvee_{i \in \mathcal{I}_j} \begin{bmatrix} W_{ij}(d) \\ r_{ij}(Dy, y(d), z, d) \leqslant 0 \end{bmatrix}, \quad d \in \mathcal{D}, j \in \mathcal{J}$$

$$\Omega(W(d)) = \text{True}, \qquad d \in \mathcal{D}$$

$$W_{ij}(d) \in \{\text{True}, \text{False}\}, \qquad d \in \mathcal{D}, i \in \mathcal{I}_j, j \in \mathcal{J}$$

- `DisjunctiveProgramming.jl` automates reformulation
- `InfiniteOpt.jl` automates transcription
- Techniques are adapted to the infinite setting

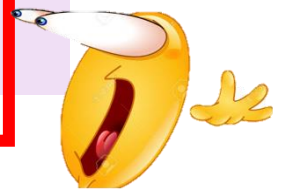
```julia
using DisjunctiveProgramming, InfiniteOpt, HiGHS

# Create the model

# Create the infinite variables



# Add the disjunctions and their indicator variables



# Add the logical propositions

# Reformulate and solve
```



WHY INFINITEDISJUNCTIVEPROGRAMMING.JL?

- InfiniteDisjunctiveProgramming.jl adapts techniques for you
- Others like PyomoDAE do not bridge GDP and InfiniteOpt

	PYOMO GDP	GAMS	DisjunctiveProgramming.jl
GDP Support	✓	✓	✓
InfiniteGDP	✗	✗	✓
Free	✓	✗	✓



CASE STUDY:
SIMULATING SILVERSTONE CIRCUIT

Using `InfiniteDisjunctiveProgramming.jl`

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

BACKGROUND

- 50/50 battery to fuel split in 2026
- **State of charge** is a real concern

4 Modes ➔ 4 disjuncts over space/time

- **Regeneration**: Recharging battery through brake
- **Internal Combustion Engine** (ICE): Using fuel
- **Boost**: Electric + ICE
- **Clipping**: ICE + Charging Battery

Photo by Ben Sutherland CC BY-SA 2.0

MATH AND CODE

$$\min \quad \int_0^L \frac{dt}{ds}\, ds$$

$$\text{s.t.} \quad v(s)\,\tfrac{dt}{ds} = 1$$

$$m\,v(s)\,\tfrac{dv}{ds} = F_e(s) - \tfrac{1}{2}\rho\, C_d A\, v(s)^2 - f_r\, m\, g$$

$$E_b\, v(s)\,\tfrac{d\,\mathrm{SOC}}{ds} = -P_b(s)$$

$$P_m(s) = F(s)\, v(s)$$

$$\bar{v}(\kappa) = \sqrt{\frac{\mu\, g}{\kappa - \frac{\mu\,\rho\, C_L A}{2m}}}$$

$$v_{\min} \leqslant v(s) \leqslant \min\{v_{\max},\ \bar{v}(\kappa(s))\}$$

$$\mathrm{SOC}_{\min} \leqslant \mathrm{SOC}(s) \leqslant \mathrm{SOC}_{\max}$$

$$\bigvee_{i \in \{\text{clip, ICE, boost, regen}\}} \begin{bmatrix} Y_i(s) \\ \underline{F}_i \leqslant F(s) \leqslant \bar{F}_i \\ \underline{P}_m^i \leqslant P_m(s) \leqslant \bar{P}_m^i \\ F_e(s) = \eta_{\mathrm{dt}}^i\, F(s) \\ \underline{P}_b^i \leqslant P_b(s) \leqslant \bar{P}_b^i \\ g_i\big(P_m(s),\, P_b(s)\big) \leqslant 0 \end{bmatrix}$$

$$\mathrm{EXACTLY}\big(1,\ Y_{\text{clip}}(s),\ Y_{\text{ice}}(s),\ Y_{\text{boost}}(s),\ Y_{\text{regen}}(s)\big)$$

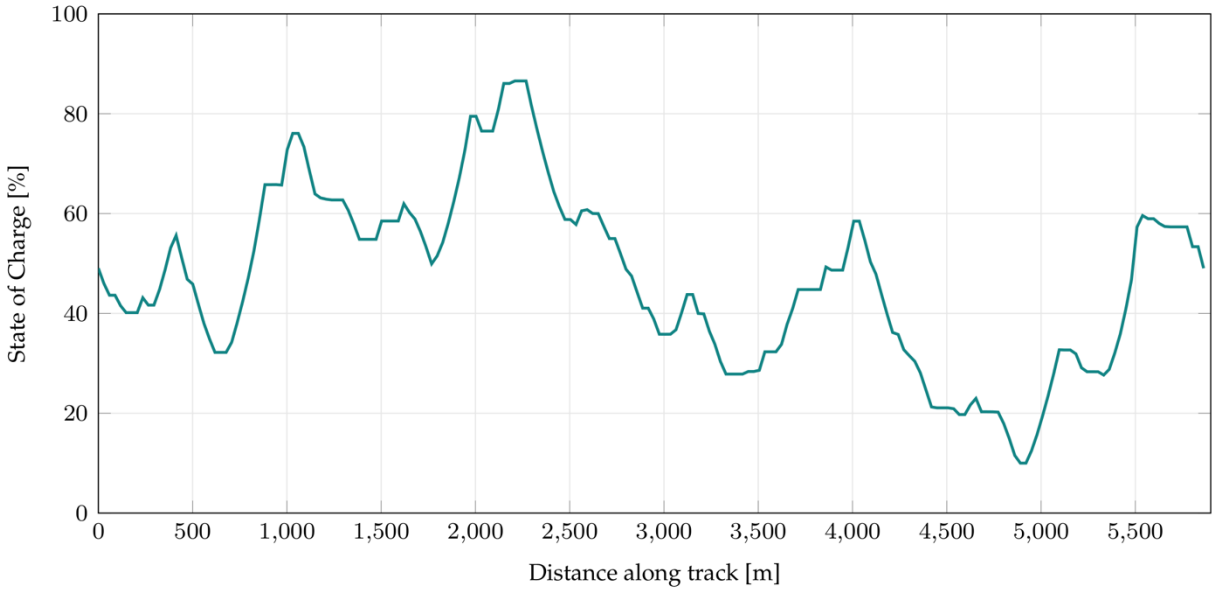
$$v(0) = v(L), \quad \mathrm{SOC}(0) = \mathrm{SOC}(L), \quad t(0) = 0$$

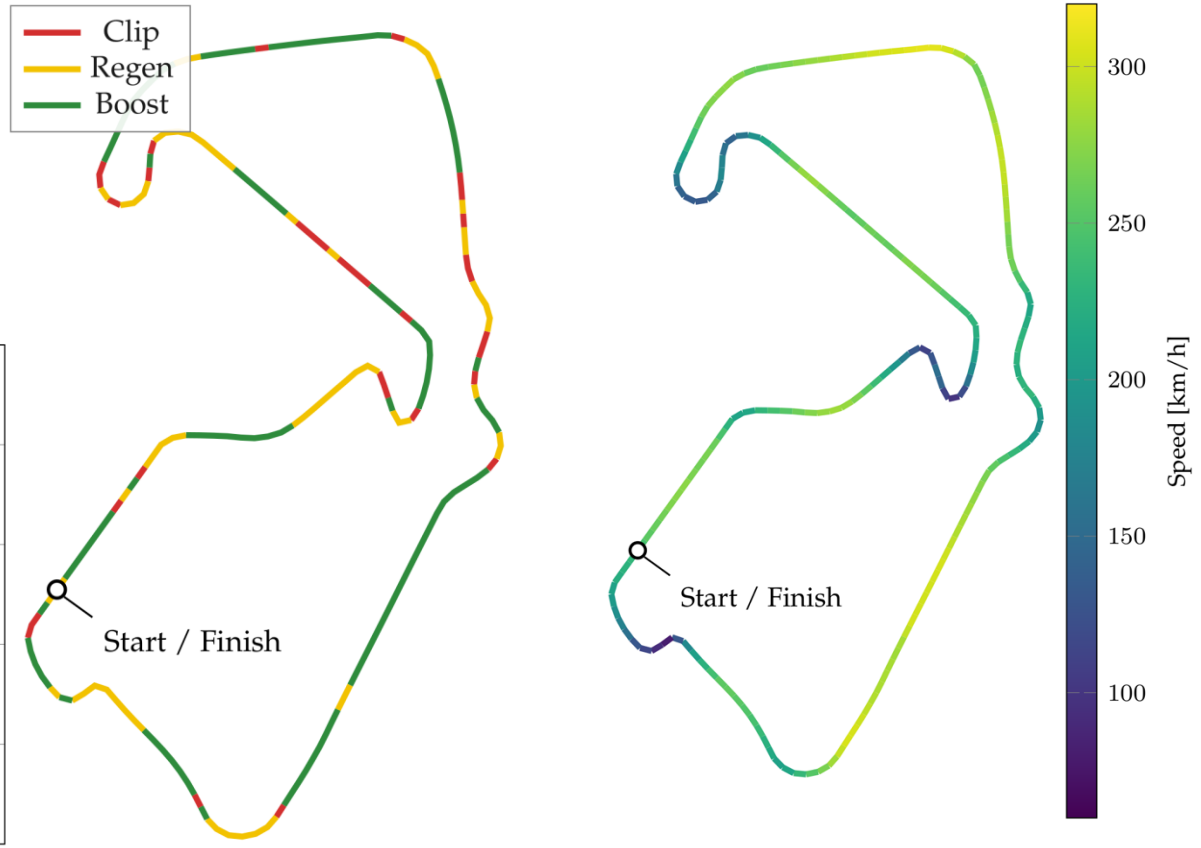
$$\forall\, s \in [0, L]$$

```
model = InfiniteGDPModel(Gurobi.Optimizer)

@infinite_parameter(model, s in [0, L], num_supports = N)
@objective(model, Min, integral(deriv(t, s), s))
@constraint(model, v * deriv(t, s) == 1)

@constraint(model, m_car * v * deriv(v, s) == F_eff - C_drag * v^2 -
    ↪ C_roll)
@constraint(model, E_batt * v * deriv(SOC, s) == -P_batt)
@constraint(model, P_motor == F_motor * v)

v_max(s) = sqrt(mu * g / (kappa(s) - mu * rho * ClA / (2 * m_car)))
@variable(model, v_min_global ⩽ v    ⩽ v_max_global, Infinite(s))
@constraint(model, v ⩽ v_max_pf)
@variable(model, SOC_min      ⩽ SOC  ⩽ SOC_max,      Infinite(s))

@variable(model, Y_clip,  InfiniteLogical(s))
@variable(model, Y_ice,   InfiniteLogical(s))
@variable(model, Y_boost, InfiniteLogical(s))
@variable(model, Y_regen, InfiniteLogical(s))
...
@constraint(model, F_motor ⩾ F_motor_min_engine, Disjunct(Y_ice))
@constraint(model, P_motor ⩽ P_ICE_max,          Disjunct(Y_ice))
...
@disjunction(model, [Y_clip, Y_ice, Y_boost, Y_regen], exactly1 =
    ↪ true)
@constraint(model, v(0)   == v(L))
@constraint(model, SOC(0) == SOC(L))
@constraint(model, t(0)   == 0)

optimize!(model, gdp_method = MBM(Gurobi.Optimizer))
```

Results

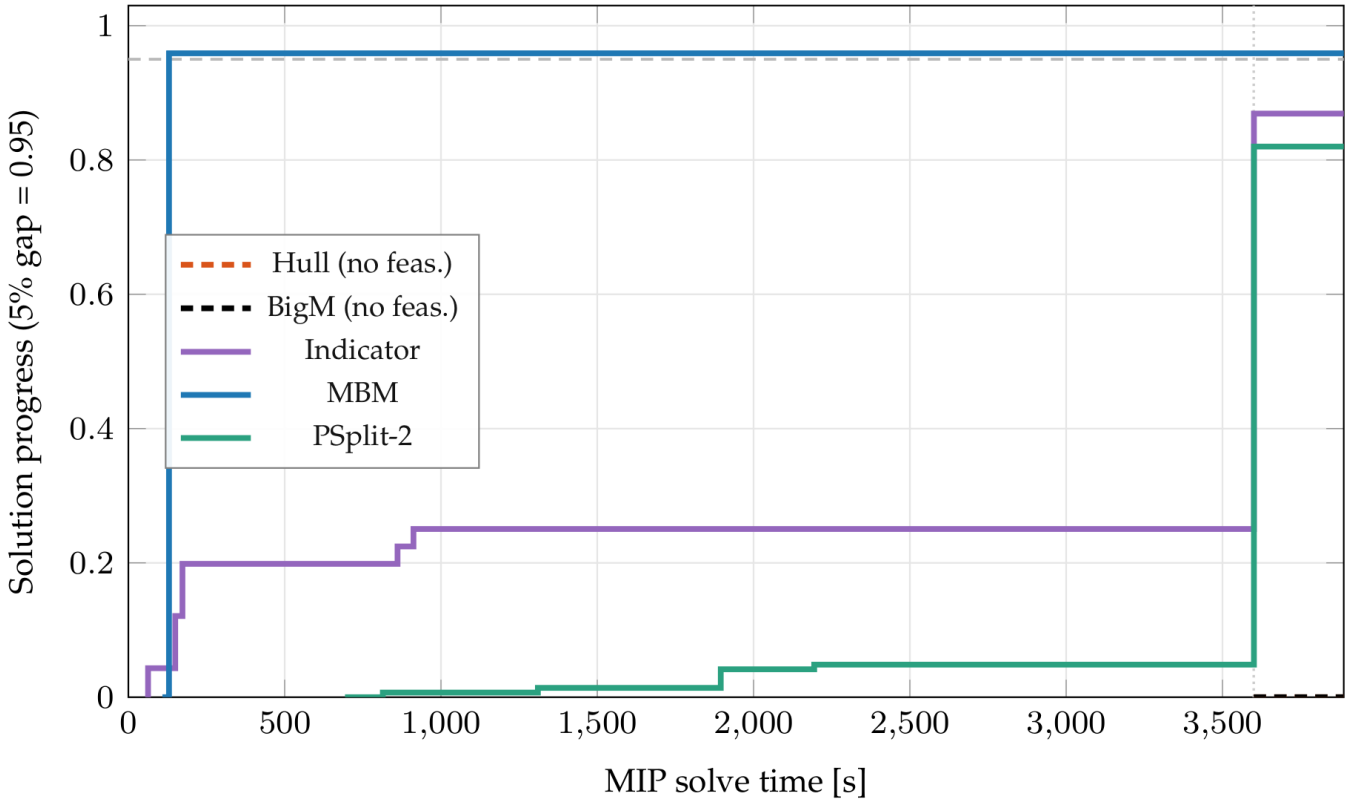
- State of charge changes over time
- Mode selection is intuitively sound
- One decision (ICE) proved to be redundant





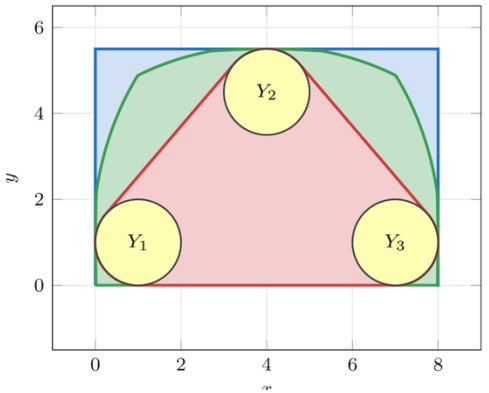
METHOD SOLVE TIMES

- **Hull:** No feasible solution found
- **Big-M:** No feasible solution found
- **Indicator:** 13.1% gap in 1 hour
- NEW **Multiple Big-M:** Optimal in ~**2 minutes**
- NEW **P-Split:** 18.0% gap in 1 hour
- NEW ~~**Cutting Planes:** Inappropriate technique~~



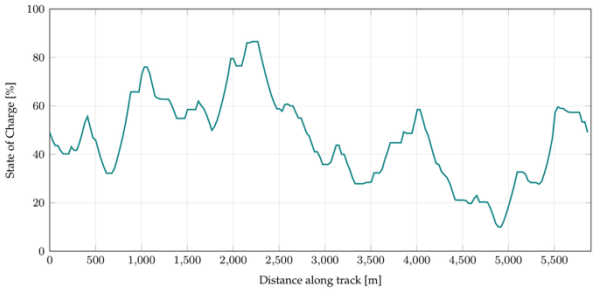
RECAP

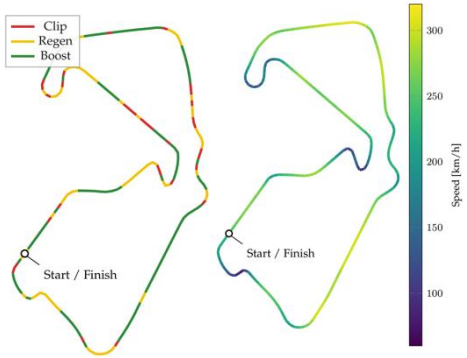
- Big M and Hull are extremes of MIP relaxation
- Other methods exist to compromise
- Method switching is tedious
- DisjunctiveProgramming.jl automates GDP to MIP
- InfiniteDisjunctiveProgramming enables dynamic GDP




```
# --- 2. Pick a reformulation method ---
optimize!(model, gdp_method = BigM())
```

DisjunctiveProgramming.jl + InfiniteOpt





UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

YOU+WATERLOO

Our greatest impact happens together.