THE NEW: DisjunctiveProgramming.jl

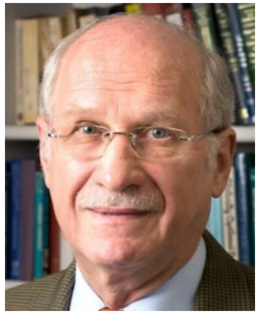
7/29/2024

Joshua Pulsipher and Hector Perez (RelationalAI)

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

ACKNOWLEDGEMENTS

Carl Laird
CMU
Professor



Ignacio Grossmann
CMU
Professor

Hector Perez
RelationalAI
Researcher

UNIVERSITY OF
WATERLOO
Department of
Chemical Engineering

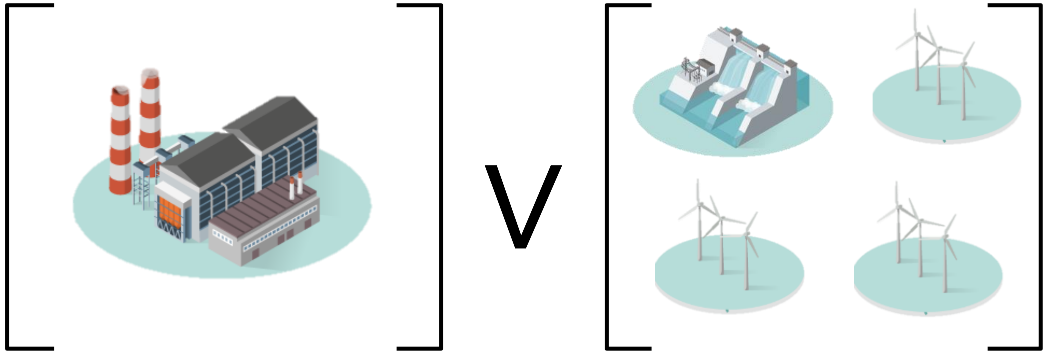
CAPD
Carnegie Mellon University
Center for Advanced
Process Decision-making

OUTLINE

- Background
- Modelling API
- Solution Approaches
- Infinite GDP

OUTLINE

- **Background**
- Modelling API
- Solution Approaches
- Infinite GDP



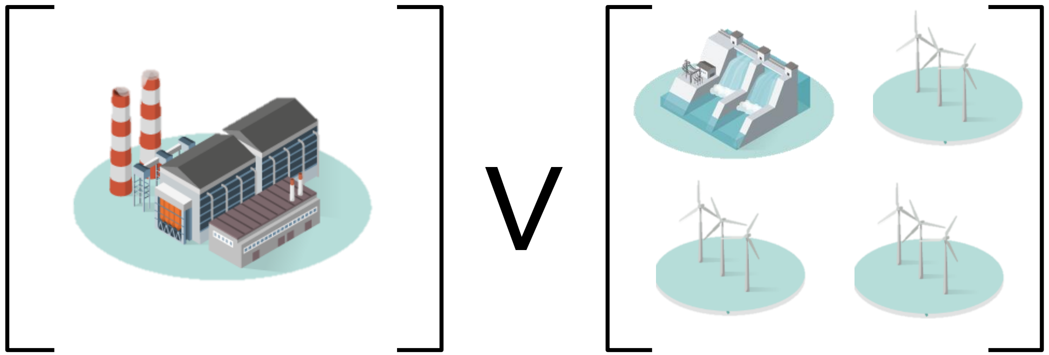
WHAT IS GENERALIZED DISJUNCTIVE PROGRAMMING?

Disjunctions

- **Conditionally enforce constraints** based on values of Boolean variables $G \in [True, False]$

$$\bigvee_{j \in \mathcal{J}_i} \begin{bmatrix} G_{ij} \\ g_{ij}(z) \leq 0 \end{bmatrix}, \quad i \in \mathcal{I}$$

- **Example:** Expansion Planning



Logic Constraints

- Enforce logic on Boolean variables G

$$\Omega(G) = True$$

- Propositional logic

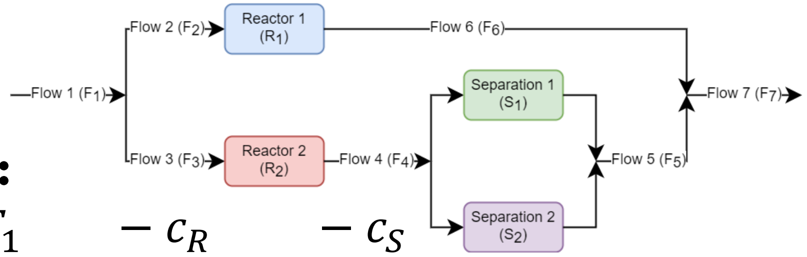
$$\wedge \qquad \vee \qquad \neg \qquad \Longrightarrow \qquad \Longleftrightarrow$$

- Constraint programming logic

$$\text{atleast}(n, \cdot) \qquad \text{exactly}(n, \cdot) \qquad \text{atmost}(n, \cdot)$$

- **Systematic conversion** to algebraic constraints
  - Apply De Morgan's laws to convert to conjunctive normal form

THE OLD



Objective Function:

$$\max \; p_7 \cdot F_7 \quad - p_1 \cdot F_1 \quad - c_R \quad - c_S$$

Global Constraints: *(Material Balances)*

$$F_1 = F_2 + F_3$$
$$F_7 = F_5 + F_6$$

Disjunctions: *(Technology Selection)*

$$\begin{bmatrix} Y_{R_1} \\ F_6 = \beta_{R_1} \cdot F_2 \\ F_3 = F_4 = F_5 = 0 \\ c_R = \gamma_{R_1} \\ c_S = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_{R_2} \\ F_6 = F_2 = 0 \\ F_4 = \beta_{R_2} \cdot F_3 \\ c_R = \gamma_{R_2} \\ \left[\begin{bmatrix} Y_{S_1} \\ F_5 = \beta_{S_1} \cdot F_4 \\ c_S = \gamma_{S_1} \end{bmatrix} \bigvee \begin{bmatrix} Y_{S_2} \\ F_5 = \beta_{S_2} \cdot F_4 \\ c_S = \gamma_{S_2} \end{bmatrix} \right] \end{bmatrix}$$

Logic Constraints: *(Technology Selection)*

$$Y_{R_1} \veebar Y_{R_2}$$
$$Y_{R_2} \Leftrightarrow \left(Y_{S_1} \veebar Y_{S_2}\right) \quad \leftarrow$$
$$Y_{R_1} \Leftrightarrow \left(\neg Y_{S_1} \wedge \neg Y_{S_2}\right) \quad \leftarrow$$

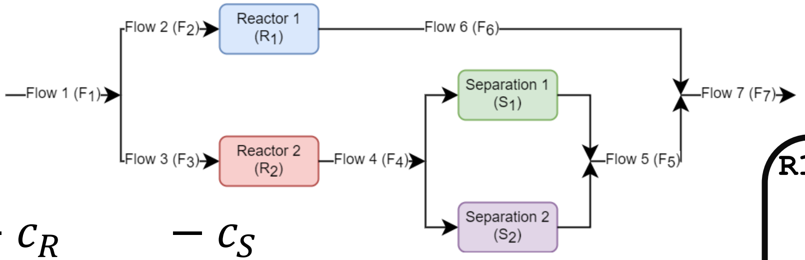
Alternate Propositions

```julia
using JuMP, DisjunctiveProgramming, Random

#parameters
p = Dict(1 => rand(), 7 => 1 + rand())
β = Dict(r => rand() for r in [:R1, :R2])
γ = Dict(r => rand() for r in [:R1, :R2, :S1, :S2])
γSmax = maximum(γ[:S1], γ[:S2])
γRmax = maximum(γ[:R1], γ[:R2])
γRmin = minimum(γ[:R1], γ[:R2])

m = Model()
@variable(m, 0 ≤ F[i = 1:7] ≤ 10)
@variable(m, 0 ≤ CS ≤ γSmax)
@variable(m, γRmin ≤ CR ≤ γRmax)
@constraints(m,
    begin
        F[1] == F[2] + F[3]
        F[7] == F[5] + F[6]
    end
)
inner_disj = @disjunction(m,
    begin
        F[5] == β[:S1]*F[4]
        CS == γ[:S1]
    end,
    begin
        F[5] == β[:S2]*F[4]
        CS == γ[:S2]
    end,
    reformulation = :big_m,
    name = :YS
)
```

← *- Reformulation type*

← *- Indicator Variable*

THE OLD



Objective Function:

$$\max \ p_7 \cdot F_7 - p_1 \cdot F_1 - c_R - c_S$$

Global Constraints: *(Material Balances)*

$$F_1 = F_2 + F_3$$
$$F_7 = F_5 + F_6$$

Disjunctions: *(Technology Selection)*

$$\begin{bmatrix} Y_{R_1} \\ F_6 = \beta_{R_1} \cdot F_2 \\ F_3 = F_4 = F_5 = 0 \\ c_R = \gamma_{R_1} \\ c_S = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_{R_2} \\ F_6 = F_2 = 0 \\ F_4 = \beta_{R_2} \cdot F_3 \\ c_R = \gamma_{R_2} \\ \begin{bmatrix} Y_{S_1} \\ F_5 = \beta_{S_1} \cdot F_4 \\ c_S = \gamma_{S_1} \end{bmatrix} \bigvee \begin{bmatrix} Y_{S_2} \\ F_5 = \beta_{S_2} \cdot F_4 \\ c_S = \gamma_{S_2} \end{bmatrix} \end{bmatrix}$$

Logic Constraints: *(Technology Selection)*

$$Y_{R_1} \underline{\vee} Y_{R_2}$$
$$Y_{R_2} \Leftrightarrow \left(Y_{S_1} \underline{\vee} Y_{S_2}\right)$$
$$Y_{R_1} \Leftrightarrow \left(\neg Y_{S_1} \wedge \neg Y_{S_2}\right)$$

Alternate Propositions

```julia
R1_con = @constraints(m,
    begin
        F[6] == β[:R1]*F[2]
        F[3] == 0
        F[4] == 0
        F[5] == 0
        CR == γ[:R1]
        CS == 0
    end
)
R2_con = @constraints(m,
    begin
        F[6] == β[:R2]*F[3]
        CR == γ[:R2]
    end
)
add_disjunction!(m,
    R1_con,
    (
        R2_con,
        values(inner_disj)...
    ),
    reformulation = :big_m,
    name = :YR
)
choose!(1, YR[1], YR[2]; mode = :exactly)
choose!(YR[2], YS[1], YS[2]; mode = :exactly)
#@proposition(m, YR[1] ⇔ (¬YS[1] ∧ ¬YS[2]))
```

OLD LIMITATIONS (FROM LAST JUMP-DEV)

- Syntax doesn’t closely match mathematical representation
- Cannot change transformation and resolve
- Doesn’t scale for a large # of disjunctions
- Not compatible with InfiniteOpt.jl
- Doesn’t support nonlinear expressions (due to current JuMP limitations)

OUTLINE

- Background
- **Modelling API**
- Solution Approaches
- Infinite GDP

```
using DisjunctiveProgramming, HiGHS
model = GDPModel(HiGHS.Optimizer)

# Variables
@variable(model, z[j ∈ 1:10])
@variable(model, Y[1:2, 1:10], Logical)
@variable(model, W[1:10], Logical)

# Disjunct 1
@constraint(model, [j ∈ 1:10], 3z[j]^2 <= 4, Disjunct(Y[1, j]))
@constraint(model, [j ∈ 1:10], sin(z[j]) == 42, Disjunct(Y[1, j]))
# Disjunct 2
@constraint(model, [j ∈ 1:10], 2z[j] <= 1, Disjunct(Y[2, j]))
# Disjunction
@disjunction(model, obj_name[j ∈ 1:10], Y[:, j])

# Logic constraints
@constraint(model, [j ∈ 1:10], ¬Y[1, j] ∧ Y[2, j] ⟹ W[j] := true)

# Cardinality constraints
@constraint(model, [j ∈ 1:10], Y[:, j] in AtLeast(1))
```

LOGICAL VARIABLES

- Defined on the set $\{False, True\}$
- Used as indicator for disjuncts and to build logical constraints
- Syntax

```
using DisjunctiveProgramming, HiGHS

model = GDPModel(HiGHS.Optimizer) # wraps a JuMP model

@variable(model, Y[1:2], Logical) # just add the `Logical` tag
```

DISJUNCTIONS

- Disjuncts are identified via an associated logical variable (i.e., the indicator)
- Add constraints to a disjunct via the Disjunct tag
- Create disjunctions w/ @disjunction using the indicator variables

```julia
using DisjunctiveProgramming, HiGHS
model = GDPModel(HiGHS.Optimizer) # wraps a JuMP model
@variable(model, z[j ∈ 1:10])
@variable(model, Y[1:2, 1:10], Logical) # just add the `Logical` tag

# Disjunct 1
@constraint(model, [j ∈ 1:10], 3z[j]^2 <= 4, Disjunct(Y[1, j]))
@constraint(model, [j ∈ 1:10], sin(z[j]) == 42, Disjunct(Y[1, j]))
# Disjunct 2
@constraint(model, [j ∈ 1:10], 2z[j] <= 1, Disjunct(Y[2, j]))
# Disjunction
@disjunction(model, obj_name[j ∈ 1:10], Y[:, j])
```

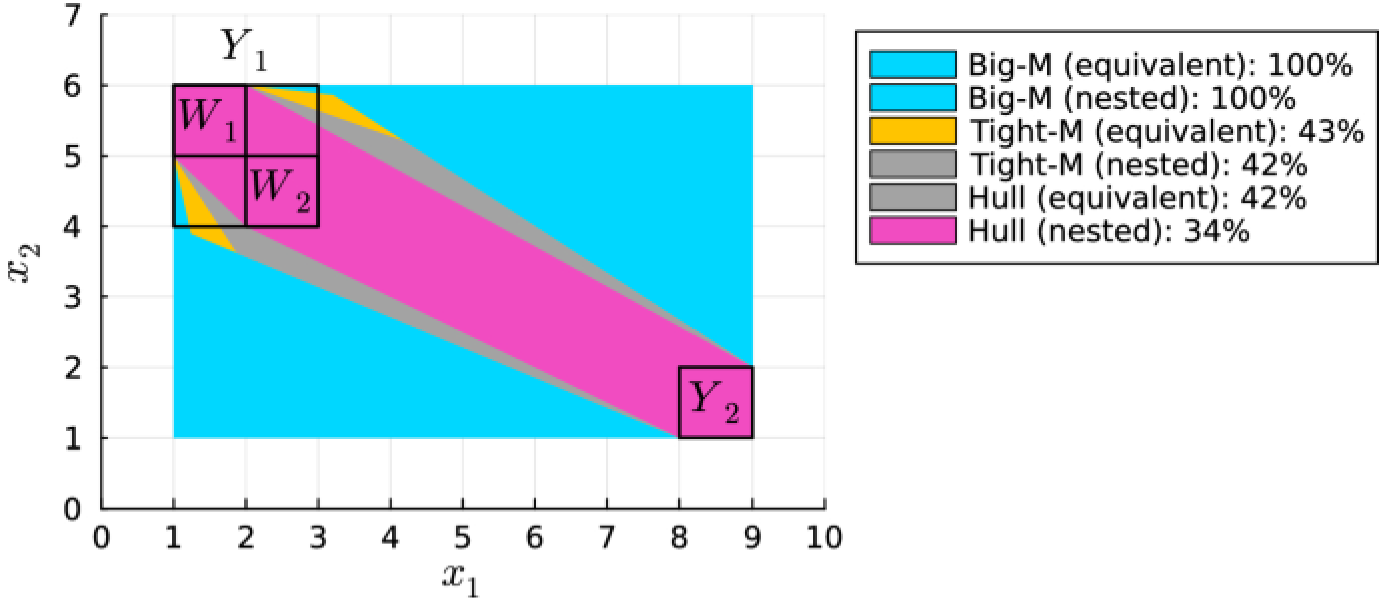
LOGIC CONSTRAINTS

- Supported logical operators: ∨ (||), ∧ (&&), ¬, ⟹, and ⇔ (==)
- Use JuMP's Boolean constraint syntax using only logical variables
- Also supports cardinality constraints via AtMost, AtLeast, & Exactly sets

```
using DisjunctiveProgramming, HiGHS
model = GDPModel(HiGHS.Optimizer)
@variable(model, Y[1:2, 1:10], Logical)
@variable(model, W[1:10], Logical)

@constraint(model, [j ∈ 1:10], ¬Y[1, j] ∧ Y[2, j] ⟹ W[j] := true)

@constraint(model, [j ∈ 1:10], Y[:, j] in AtLeast(1))
```

OUTLINE

- Background
- Modelling API
- **Solution Approaches**
- Infinite GDP



DISJUNCTION REFORMULATIONS

- **Idea:** Convert logic variables to binary & then reformulate disjunctions to MIP
- Reformulates the JuMP model directly (reformulations can be undone too)
- Currently supported reformulations
  - Big-M
  - (Convex) Hull
  - Indicator Constraints

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

- Minimal extension API to add more

LOGIC CONSTRAINT REFORMULATIONS

- Logical constraints are reformulated automatically into (MI)LP constraints
- Accomplishes this by first converting to conjunctive normal form
- Selecting a different approach is not currently supported

```
optimize!(model)
```

FUTURE DEVELOPMENT PLANS

- Create MOI objects and solver to apply more specialized approaches (e.g., LOA)

- Add more reformulation techniques (e.g., multiple big-M, P-split)

OUTLINE

- Background
- Modelling API
- Solution Approaches
- **Infinite GDP**

Open [DO NOT MERGE YET] Add InfiniteOpt as an Extension #114

pulsipher wants to merge 4 commits into `master` from `infiniteopt_ext`

```julia
using DisjunctiveProgramming, InfiniteOpt, HiGHS

# Create the model
model = InfiniteGDPModel(HiGHS.Optimizer)

# Create the infinite variables
I = 1:4
@infinite_parameter(model, t ∈ [0, 1], num_supports = 100)
@variable(model, 0 <= g[I] <= 10, Infinite(t))

# Add the disjunctions and their indicator variables
@variable(model, G[I, 1:2], InfiniteLogical(t))
@constraint(model, [i ∈ I, j ∈ 1:2], 0 <= g[i], Disjunct(G[i, 1]))
@constraint(model, [i ∈ I, j ∈ 1:2], g[i] <= 0, Disjunct(G[i, 2]))
@disjunction(model, [i ∈ I], G[i, :])

# Add the logical propositions
@variable(model, W, InfiniteLogical(t))
@constraint(model, G[1, 1] ∨ G[2, 1] ∧ G[3, 1] == W := true)
@constraint(model, 𝔼(binary_variable(W), t) >= 0.95) # incorporate bina

# Reformulate and solve
optimize!(model, gdp_method = Hull())

# check the results
value(W)
```

INFINITE-DIMENSIONAL GDP

- Enforce disjunctions and/or logical constraints with infinite-dimensional variables and/or constraints

$$\bigvee_{j \in \mathcal{J}_i} \begin{bmatrix} G_{ij}(t, x, \xi) \\ g_{ij}(z, y(t, x, \xi)) \leq 0 \end{bmatrix}, \quad i \in \mathcal{I}, t \in \mathcal{D}_t, x \in \mathcal{D}_x, \mathcal{D}_\xi$$

- *Applications:* Strategic planning, expansion planning, scheduling, more

GDP FOR CHANCE CONSTRAINTS

Infinite GDP Derivation

- Chance constraint

$$\int_{\xi \in \mathcal{D}_\xi} \mathbb{1}_{\Omega(g(\xi) \leq 0)}(\xi) p(\xi) d\xi \geq \alpha \qquad (1)$$

- **Disjunctions** over constraint satisfaction

$$\begin{bmatrix} G_i(\xi) \\ g_i(\xi) \leq 0 \end{bmatrix} \underline{\bigvee} \begin{bmatrix} \neg G_i(\xi) \\ g_i(\xi) > 0 \end{bmatrix}, \quad i \in \mathcal{I}, \ \xi \in \mathcal{D}_\xi \qquad (2)$$

- **Express** event logic via logical propositions

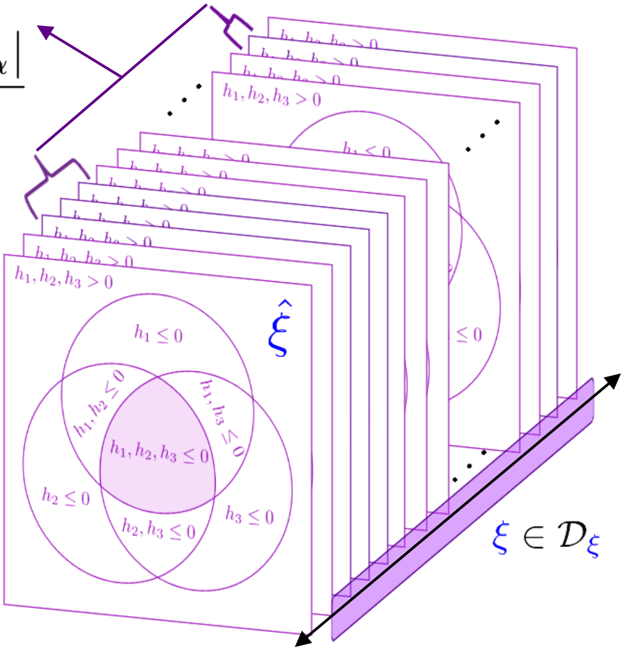
$$\begin{gathered} \Omega(G(\xi)) \iff W(\xi), \quad \xi \in \mathcal{D}_\xi \\ \mathbb{E}_\xi[W(\xi)] \geq \alpha \end{gathered} \qquad (3)$$

- **Combine** (2) and (3) to represent (1)

Illustration

- Event occurs over ***α* fraction** of ξ values*

$$\alpha \leq \frac{1 - |\mathcal{D}_{\xi,\alpha}|}{|\mathcal{D}_\xi|}$$



**Assuming constant pdf*

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

J. L. Pulsipher, D. Ovalle, H. Perez, I. E. Grossmann, C. D. Laird. "Characterizing Event Constraints with General Disjunctive Programming". 2022

MODELLING INFINITE GDPS WITH INFINITEOPT

Open **[DO NOT MERGE YET] Add InfiniteOpt as an Extension** #114

pulsipher wants to merge 4 commits into `master` from `infiniteopt_ext`

```julia
using DisjunctiveProgramming, InfiniteOpt, HiGHS

# Create the model
model = InfiniteGDPModel(HiGHS.Optimizer)

# Create the infinite variables
I = 1:4
@infinite_parameter(model, t ∈ [0, 1], num_supports = 100)
@variable(model, 0 <= g[I] <= 10, Infinite(t))

# Add the disjunctions and their indicator variables
@variable(model, G[I, 1:2], InfiniteLogical(t))
@constraint(model, [i ∈ I, j ∈ 1:2], 0 <= g[i], Disjunct(G[i, 1]))
@constraint(model, [i ∈ I, j ∈ 1:2], g[i] <= 0, Disjunct(G[i, 2]))
@disjunction(model, [i ∈ I], G[i, :])

# Add the logical propositions
@variable(model, W, InfiniteLogical(t))
@constraint(model, G[1, 1] ∨ G[2, 1] ∧ G[3, 1] == W := true)
@constraint(model, 𝔼(binary_variable(W), t) >= 0.95) # incorporate bina

# Reformulate and solve
optimize!(model, gdp_method = Hull())

# check the results
value(W)
```

- Extension that loads w/ InfiniteOpt
- Adds InfiniteGDPModel and InfiniteLogical
- More efficient reformulations

DisjunctiveProgramming.jl

UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

YOU+WATERLOO

Our greatest impact happens together.