



d|b|t|a

Process Dynamics and Operations Group

Technische
Universität
Berlin



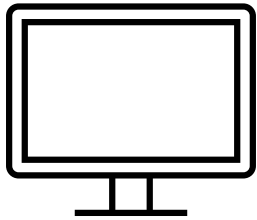
CAPE-OPEN and the Future of Superstructure Optimization

CAPE-OPEN 2024 Annual Meeting | 08.10.2024

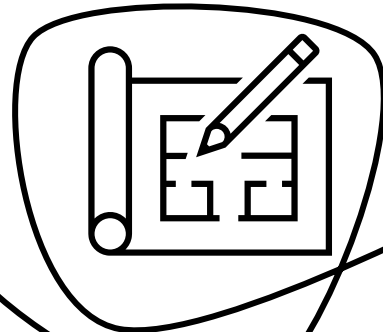
Lukas Scheffold, Erik Esche, Jens-Uwe Repke

Motivation and Background

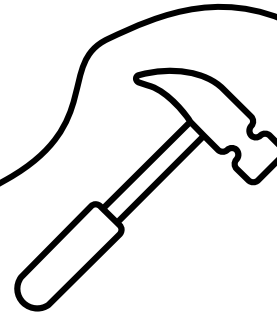
Synthesis



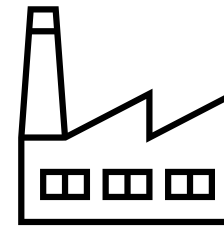
Design



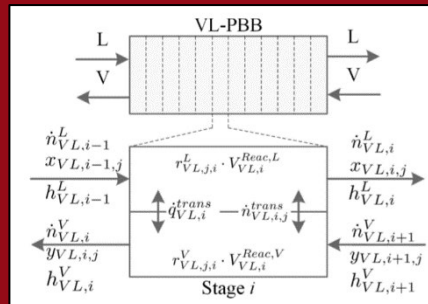
Construction



Commissioning



Motivation and Background



GENERALITY

PBB formulation in
MOSAICmodeling



FIDELITY

Rigorous thermodynamic and
kinetic models

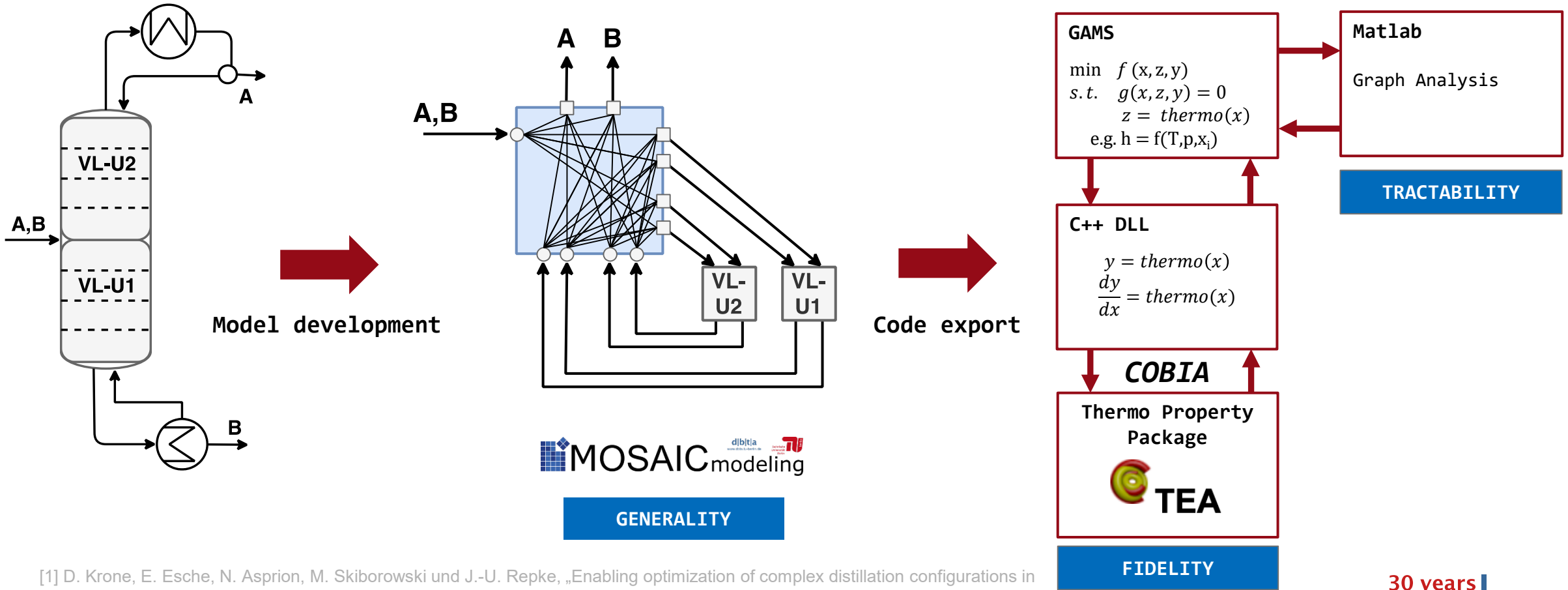


TRACTABILITY

Structural screening

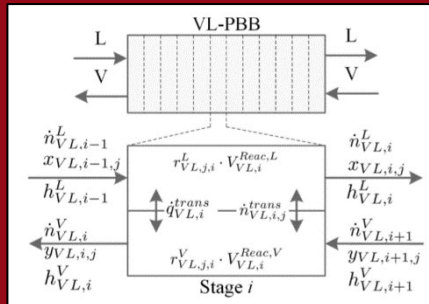
Previous Work

[1]



[1] D. Krone, E. Esche, N. Asprion, M. Skiborowski und J.-U. Repke, „Enabling optimization of complex distillation configurations in GAMS with CAPE-OPEN thermodynamic models,“ *Computers & Chemical Engineering*, Bd. 157, p. 107626, 2022.

Challenges



**Strong
Nonlinearities**

Dedicated Initialization
Procedures



Solution Time

Improvements ?



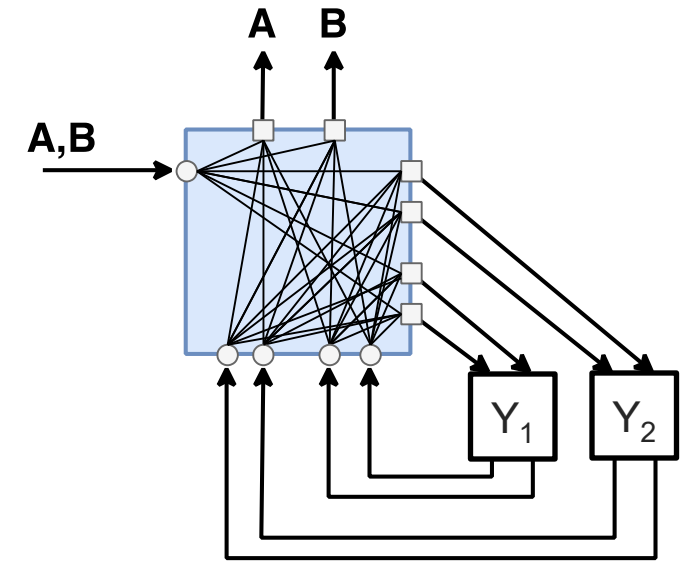
Tractability

Improvements ?

GDP Problem Formulation

[2]

| | | | |
|--------------|--|------------------------------|--------------------|
| $\min z =$ | $f(x)$ | | Objective Function |
| <i>s. t.</i> | $r(x) \leq 0$ | | Global Constraints |
| | $\bigvee_{j \in J_i} \left[\begin{array}{c} Y_{ij} \\ g_{ij}(x) \leq 0 \end{array} \right]$ | $\forall i \in I$ | Disjunctions |
| | $\Xi(1, Y_{ij} \forall j \in J_i)$ | $\forall i \in I$ | Logic Propositions |
| | $\Omega(Y)$ | | Logic Propositions |
| | $x^{LB} \leq x \leq x^{UB}$ | | Variable Bounds |
| | $x \in \mathbb{R}^n$ | | |
| | $Y_{ij} \in \{True, False\}$ | $\forall i \in I, j \in J_i$ | |

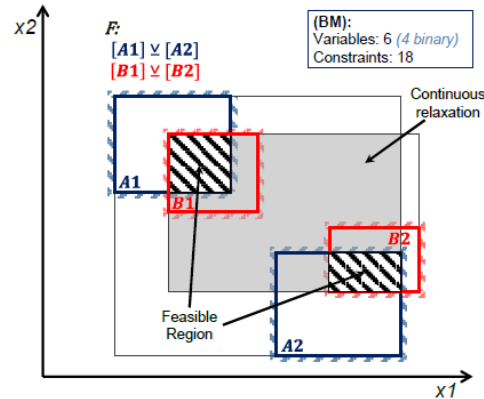


[2] Grossmann, I. E., F. Trespalacios. 2013. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal* 59 3276-3295. doi:10.1002/aic.14088. URL <http://dx.doi.org/10.1002/aic.14088>.

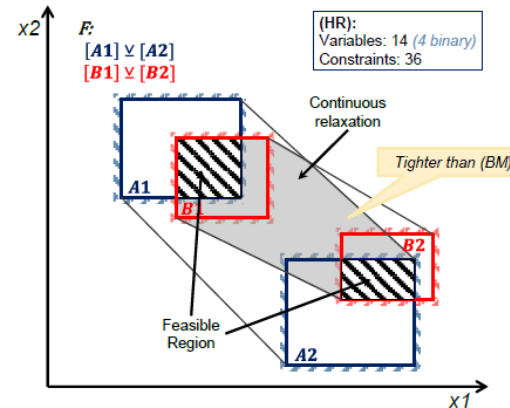
GDP Solution

[2]

Traditionally, Big M relaxation is apply by hand in MINLP formulation



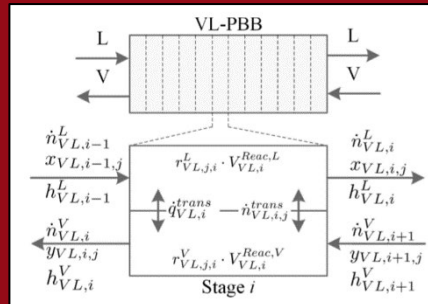
Relaxations that provide tighter bounds exist (i.e. Convex Hull relaxation).



By retaining the logical structure of the optimization problem, GDP algorithms can automatically generate/ use tighter relaxations.

[2] Grossmann, I. E., F. Trespalacios. 2013. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. AIChE Journal 59 3276-3295. doi:10.1002/aic.14088. URL <http://dx.doi.org/10.1002/aic.14088>.

Possible GDP Improvements



**Strong
Nonlinearities**

Dedicated Initialization
Procedures



Solution Time

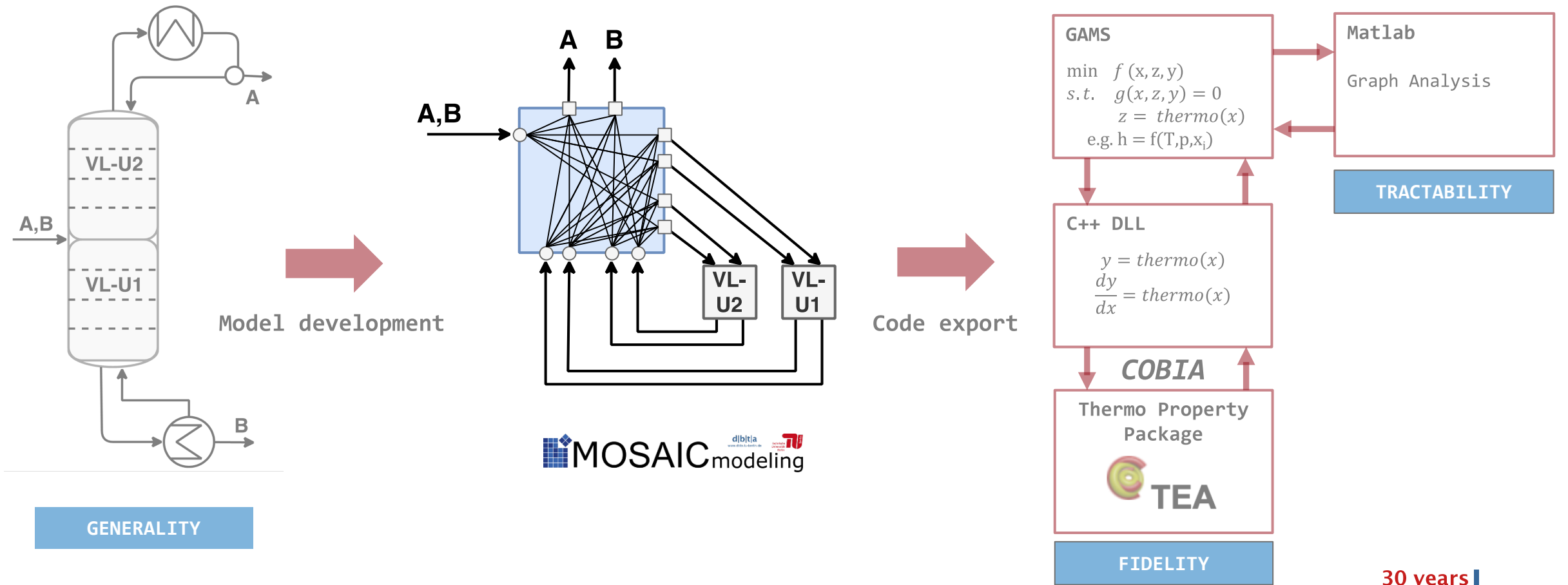
Stronger GDP Relaxations



Tractability

Additional GDP formulation

GDP Formulation in MOSAICmodeling



GENERALITY

GDP Formulation in MOSAICmodeling

Minimize $c + 2x_1 + x_2$
subject to

Objective Function

$$\left[\begin{array}{c} Y_1 \\ -x_1 + x_2 + 2 \leq 0 \\ c < 5 \end{array} \right] \vee \left[\begin{array}{c} Y_2 \\ 2 - x_2 \leq 0 \\ c < 7 \end{array} \right]$$

Disjunctions
[Equations]

$$\left[\begin{array}{c} Y_3 \\ x_1 - x_2 \leq 0 \end{array} \right] \vee \left[\begin{array}{c} \neg Y_3 \\ x_1 \leq 1 \end{array} \right]$$

$$\begin{aligned} Y_1 \wedge \neg Y_2 &\Rightarrow \neg Y_3 \\ Y_2 &\Rightarrow \neg Y_3 \\ Y_3 &\Rightarrow \neg Y_2 \end{aligned}$$

Logic
Propositions

$$\begin{aligned} 0 &\leq x_1 \leq 5 \\ 0 &\leq x_2 \leq 5 \\ c &\geq 0 \\ Y_j &\in \{\text{True}, \text{False}\}, j = 1, 2, 3 \end{aligned}$$

Continuous
Variables

Boolean
Variables



Model development

| Connected Elements | Disjunctions |
|---|--------------|
| (1) $Y_{j=1} \vee Y_{j=2}$ | |
| (2) $Y_{j=1} \wedge \neg(Y_{j=2}) \Rightarrow \neg$ | |
| (3) $Y_{j=2} \Rightarrow \neg(Y_{j=3})$ | |
| (4) $Y_{j=3} \Rightarrow \neg(Y_{j=2})$ | |
| (5) $c + 2 \cdot x_{i=1} + x_{i=2}$ | |
| (6) $-x_{i=1} + x_{i=2} + 2 \leq 0$ | $Y_{j=1}$ |

| Variable N... | Index | Type | Value | Lower Bo... | Upper Bo... | Integer | Logic Vari... |
|---------------|-------|-------------|-------|-------------|-------------|-------------------------------------|-------------------------------------|
| $Y_{j=1}$ | 0 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| $Y_{j=2}$ | 1 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| $Y_{j=3}$ | 2 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| c | 3 | UNSPECIFIED | 3 | 0 | 1 | <input type="checkbox"/> | <input type="checkbox"/> |
| $x_{i=1}$ | 4 | UNSPECIFIED | 1 | 0 | 5 | <input type="checkbox"/> | <input type="checkbox"/> |
| $x_{i=2}$ | 5 | UNSPECIFIED | 2 | 0 | 5 | <input type="checkbox"/> | <input type="checkbox"/> |

Code Export from MOSAICmodeling

| Connected Elements | | Disjunctions | |
|--------------------|---|--------------|--|
| (1) | $Y_j = 1 \vee Y_j = 2$ | | |
| (2) | $Y_j = 1 \wedge \neg(Y_j = 2) \Rightarrow \neg$ | | |
| (3) | $Y_j = 2 \Rightarrow \neg(Y_j = 3)$ | | |
| (4) | $Y_j = 3 \Rightarrow \neg(Y_j = 2)$ | | |
| (5) | $c + 2 \cdot x_{i=1} + x_{i=2}$ | | |
| (6) | $-x_{i=1} + x_{i=2} + 2 \leq 0$ | $Y_j = 1$ | |

| Variable N... | Index | Type | Value | Lower Bo... | Upper Bo... | Integer | Logic Vari... |
|---------------|-------|-------------|-------|-------------|-------------|-------------------------------------|-------------------------------------|
| $Y_j = 1$ | 0 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| $Y_j = 2$ | 1 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| $Y_j = 3$ | 2 | UNSPECIFIED | 0 | 0 | 1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| c | 3 | UNSPECIFIED | 3 | 0 | 1 | <input type="checkbox"/> | <input type="checkbox"/> |
| $x_{i=1}$ | 4 | UNSPECIFIED | 1 | 0 | 5 | <input type="checkbox"/> | <input type="checkbox"/> |
| $x_{i=2}$ | 5 | UNSPECIFIED | 2 | 0 | 5 | <input type="checkbox"/> | <input type="checkbox"/> |

Code export



```

binary variables
    e0_Y_j1
    e0_Y_j2
    e0_Y_j3

;
parameters
;
variables
    e0_c
    e0_x_i1
    e0_x_i2

;
    e0_c.l= 0.0;
    e0_c.up= 1.0E9;
    e0_c.lo= 0.0;
    e0_x_i1.l= 1.0;
    e0_x_i1.up= 5.0;
    e0_x_i1.lo= 0.0;
    e0_x_i2.l= 2.0;
    e0_x_i2.up= 5.0;
    e0_x_i2.lo= 0.0;

;
    ineq1.. e0_x_i1+e0_x_i2+2.0=1.0.0 ;
    ineq2.. e0_c=5.0 ;
    ineq3.. 2.0-e0_x_i2=1.0.0 ;
    ineq4.. e0_c=7.0 ;
    ineq5.. e0_x_i1-e0_x_i2=1.0.0 ;
    ineq6.. e0_x_i1=1.0.0 ;

Logic Equations
    logic1
    logic2
    logic3
    logic4

;
    logic1.. e0_Y_j1 or e0_Y_j2;
    logic2.. e0_Y_j1 and not e0_Y_j2 -> not
    e0_Y_j3;
    logic3.. e0_Y_j2 -> not e0_Y_j3;
    logic4.. e0_Y_j3 -> not e0_Y_j2;

$ifnot %*%m info%*/
put emp;
$onput
disjunction e0_Y_j2   ineq3 ineq4 else
disjunction e0_Y_j3   ineq5 else   ineq6
disjunction e0_Y_j1   ineq1 ineq2 else

$offput
    
```



```

from pyomo.environ import *
from pyomo.gdp import *

m = ConcreteModel()

# define variable
m.c = Var(bounds=(0, 1e9))
m.x1 = Var(bounds=(0, 5))
m.x2 = Var(bounds=(0, 5))

# Define Constraints
m.d11.c1 = Constraint(expr=-m.x1+m.x2+2<=0)
m.d11.c2 = Constraint(expr=m.c<=5)

m.d21.c1 = Constraint(expr=2-m.x2<=0)
m.d21.c2 = Constraint(expr=m.c<=7)

m.d31.c1 = Constraint(expr=m.x1-m.x2<=0)
m.d32.c1 = Constraint(expr=m.x1<=1)

# define global constraints
#m.c = Constraint(expr=m.y*((m.x+m.a)**2+m.b)=0)

# define objective
m.o = Objective(expr=m.c+2*m.x1+m.x2)

# Add the logical proposition
m.p2 = LogicalConstraint(expr=implies(land(m.d11.indicator_var, Inot(m.d21.indicator_var)),
Inot(m.d31.indicator_var)))
m.p3 = LogicalConstraint(expr=implies(m.d21.indicator_var, Inot(m.d31.indicator_var)))
m.p4 = LogicalConstraint(expr=implies(m.d31.indicator_var, Inot(m.d21.indicator_var)))

# Additional Logic Propositions for Disjunctions -> must be generated automatically or set manually
m.p1 = LogicalConstraint(expr=lor(m.d11.indicator_var, m.d21.indicator_var))
m.p5 = LogicalConstraint(expr=implies(m.d11.indicator_var, Inot(m.d21.indicator_var)))

# Define Disjunctions
m.disj1 = Disjunction()
m.disj2 = Disjunction()
m.disj3 = Disjunction()

# Define Cases (For y3/not(y3))
m.d11 = Disjunct()
m.d12 = Disjunct()

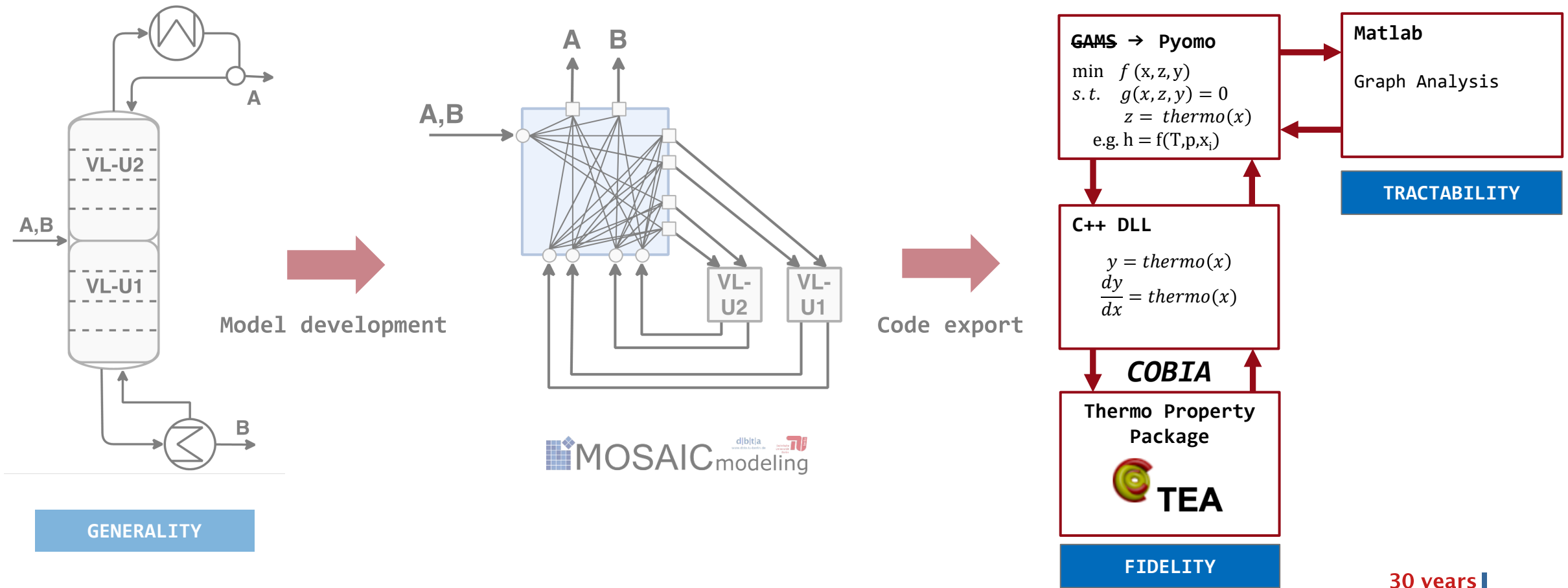
m.d21 = Disjunct()
m.d22 = Disjunct()

m.d31 = Disjunct()
m.d32 = Disjunct()

# define explicit disjunctions from cases
m.disj1 = [m.d11, m.d12]
m.disj2 = [m.d21, m.d22]
m.disj3 = [m.d31, m.d32]

# Solve the reformulated model
run_data = SolverFactory('gdpopt').solve(m, mip_solver='glpk')
    
```

Current and Future Work



Thank You For Your Attention, Questions ?

Acknowledgements



DFG

This work is funded by the Deutsche Forschungsgemeinschaft
(DFG, German Research Foundation) – 523327609.