The journey of creating a modeling language

18th INFORMS Computing Society (ICS) conference
Toronto, Canada

Bardhyl Miftari, Guillaume Derval, Quentin Louveaux, Damien Ernst
University of Liège, Belgium

This talk

Planning

Part 1:

Let’s create a modeling language

Part 2:

Go beyond state of the art

This talk

Planning

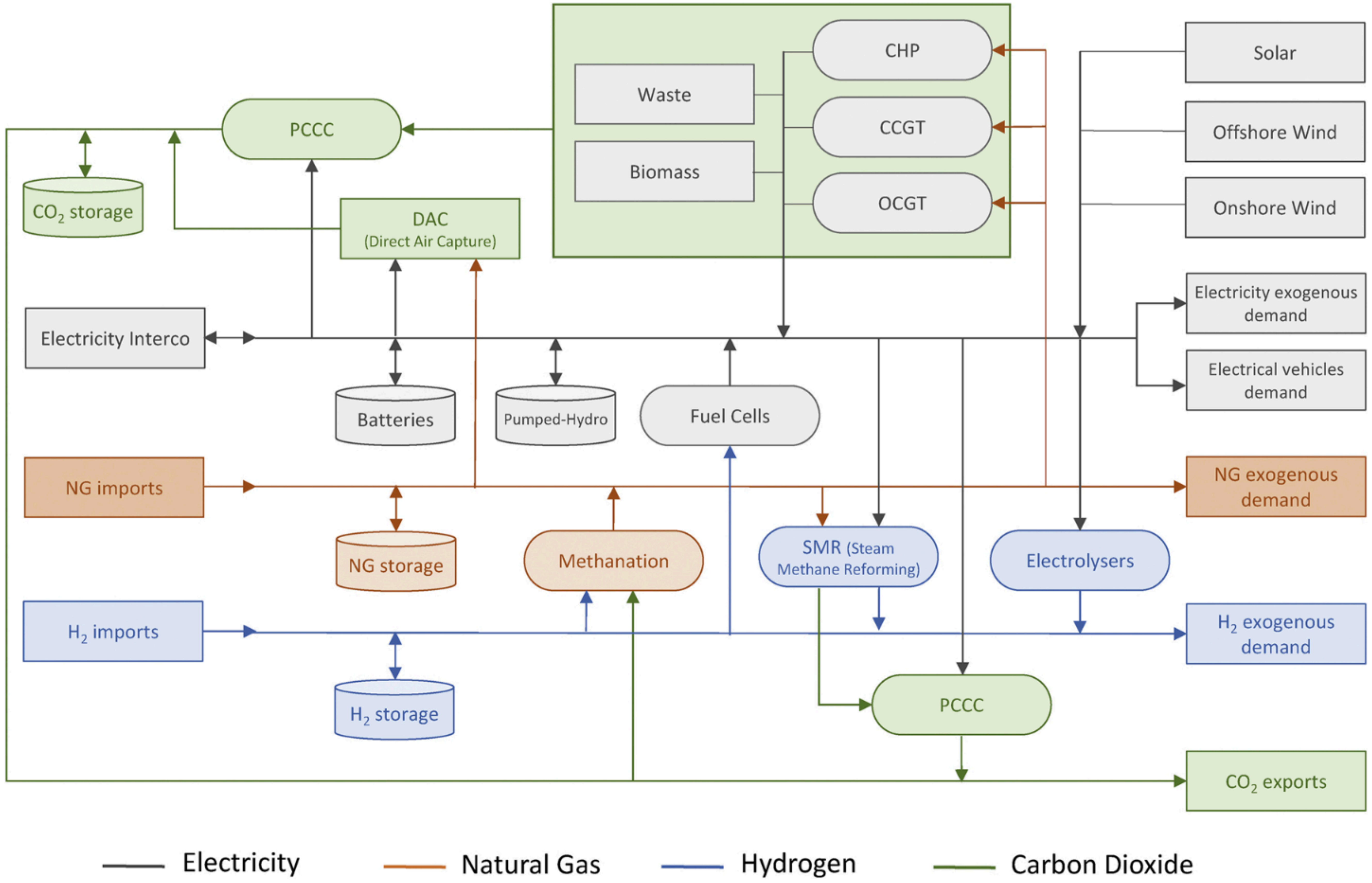
Part 1:

Let’s create a modeling language

Go back to 2021

State of the lab

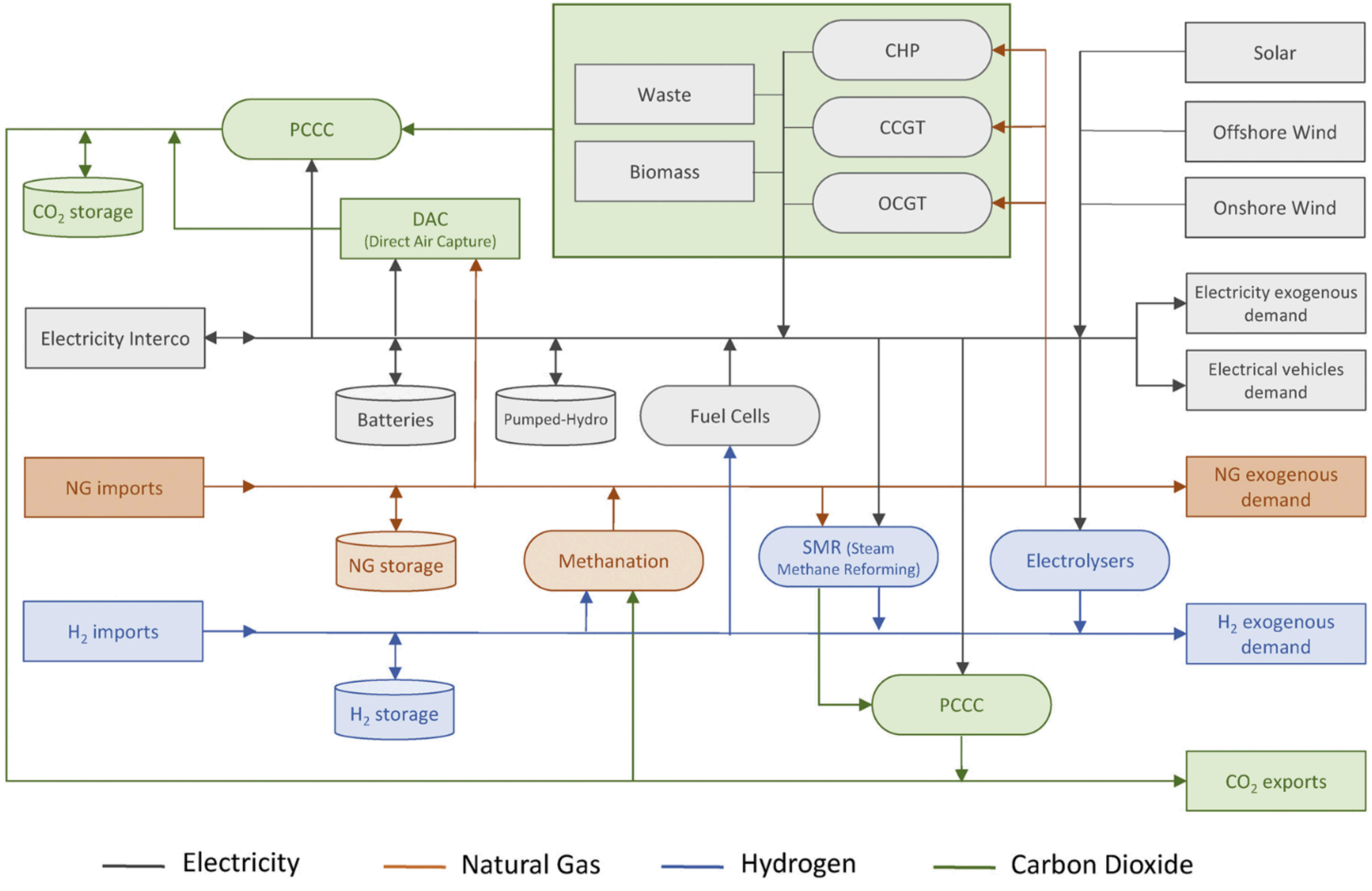
- Working on energy system planning and sizing



Go back to 2021

State of the lab

- Working on energy system planning and sizing



CHP

CCGT

OCGT

Waste

Biomass

Solar

Offshore Wind

Onshore Wind

PCCC

CO_2 storage

DAC (Direct Air Capture)

Electricity Interco

Electricity exogenous demand

Electrical vehicles demand

Batteries

Pumped-Hydro

Fuel Cells

NG imports

NG exogenous demand

NG storage

Methanation

SMR (Steam Methane Reforming)

Electrolysers

H_2 imports

H_2 exogenous demand

H_2 storage

PCCC

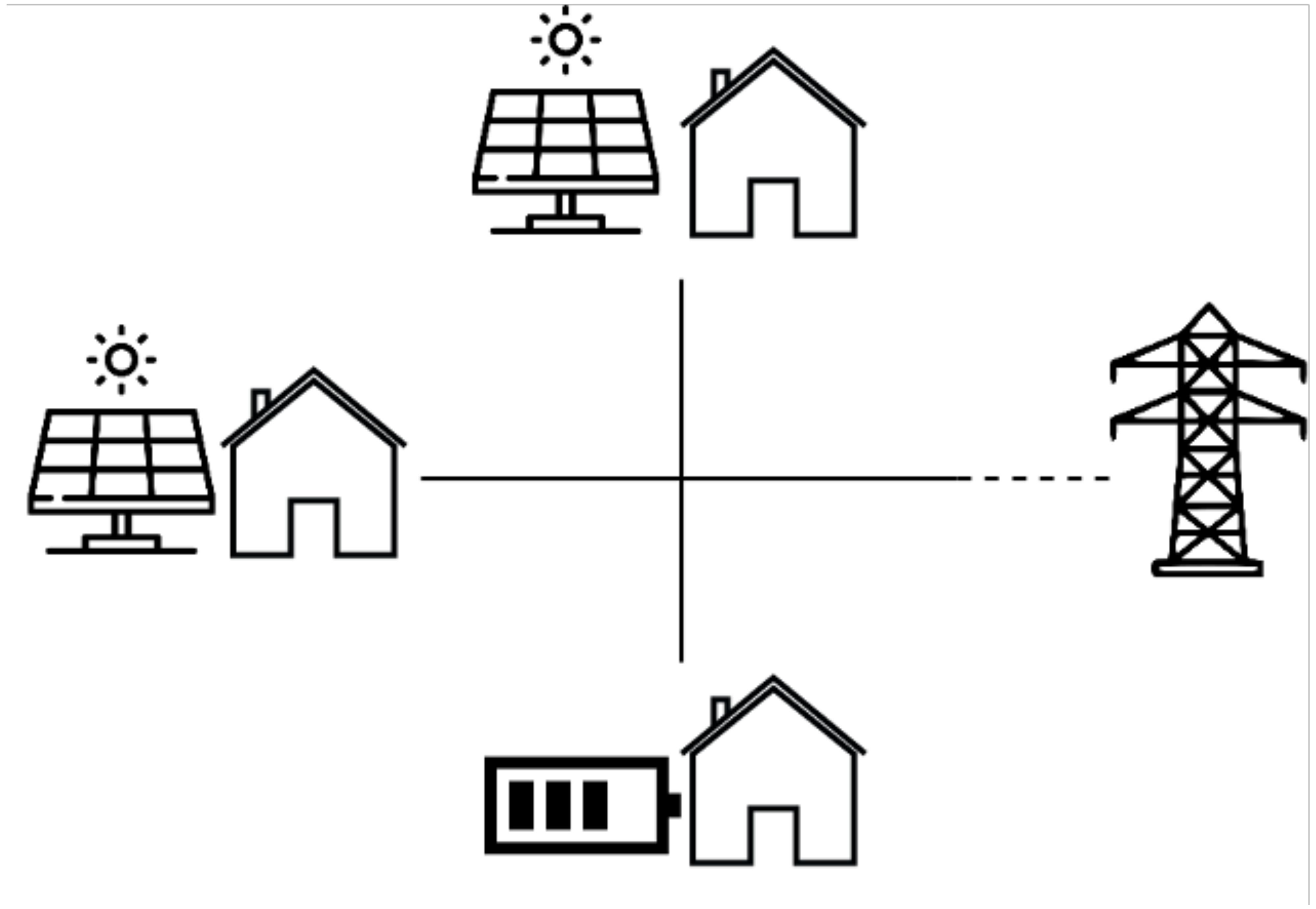
CO_2 exports

Electricity

Natural Gas

Hydrogen

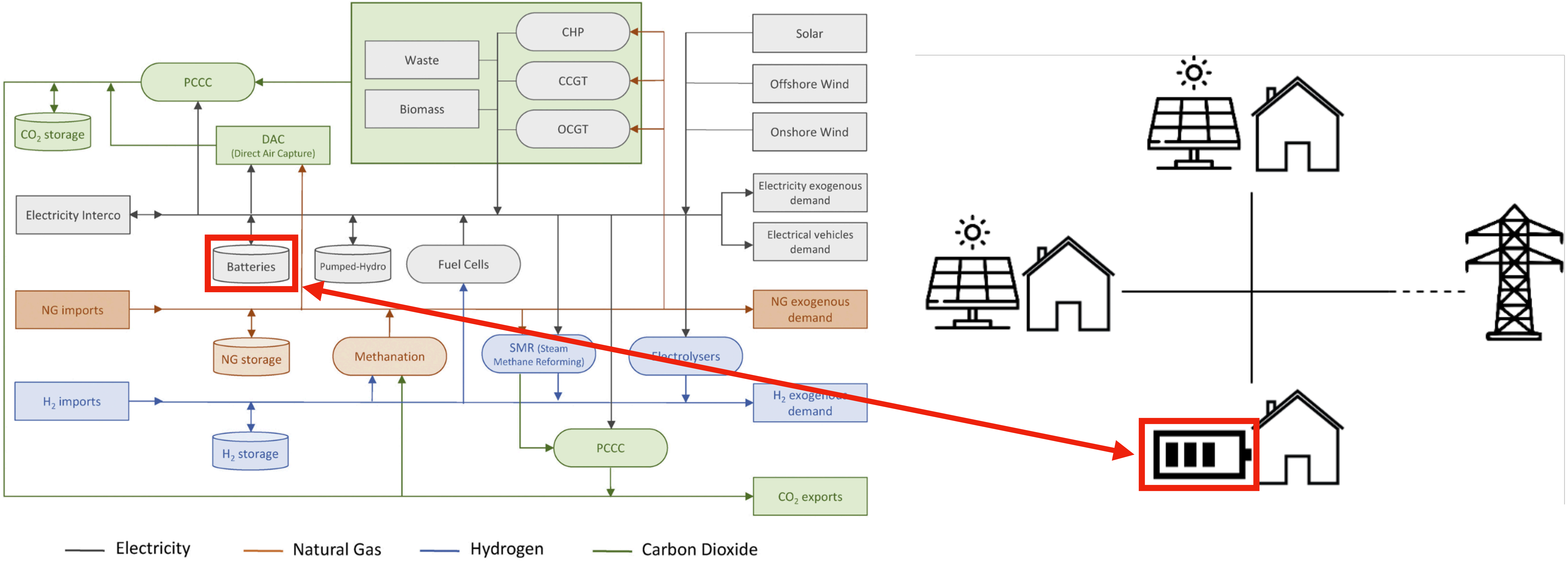
Carbon Dioxide



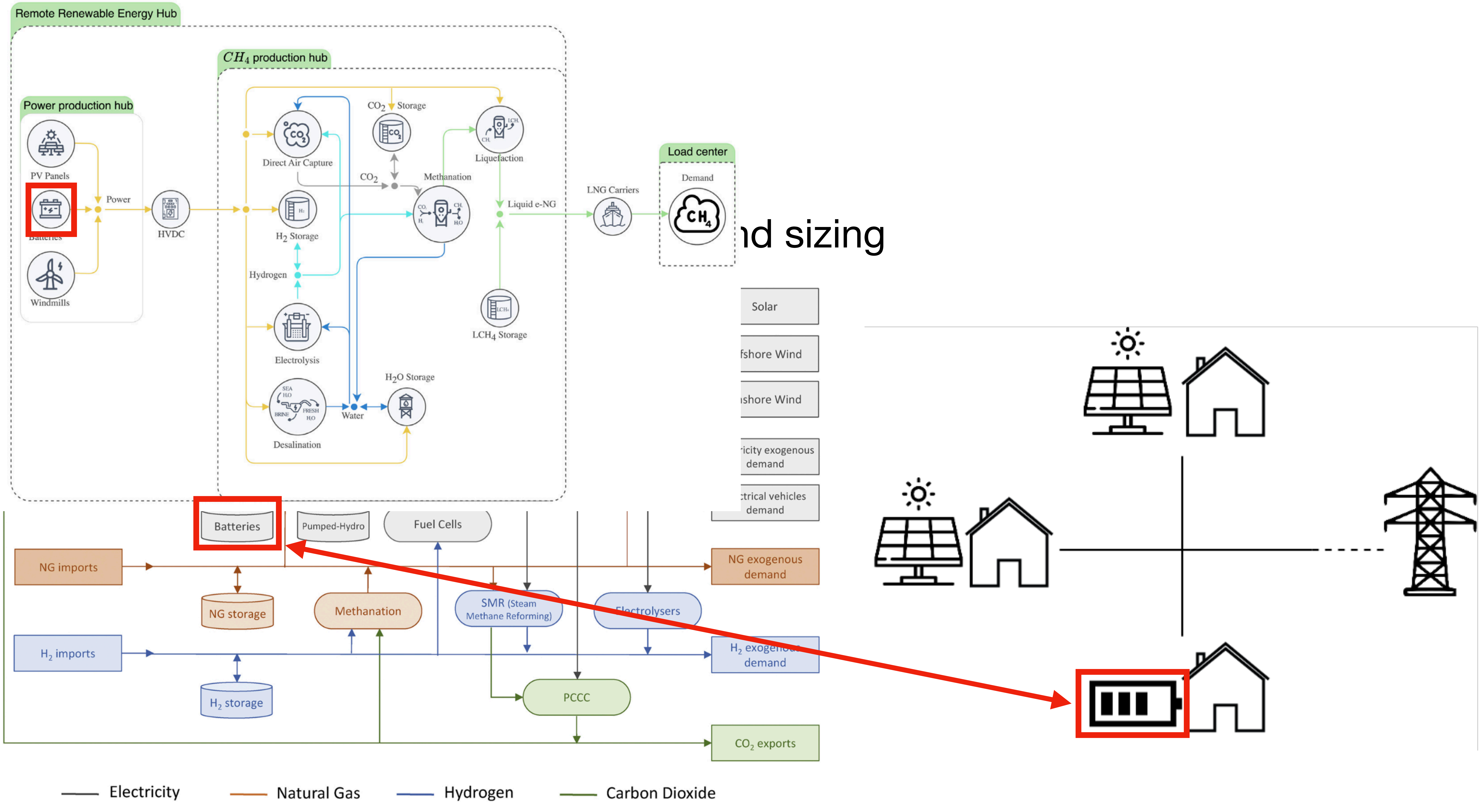
Go back to 2021

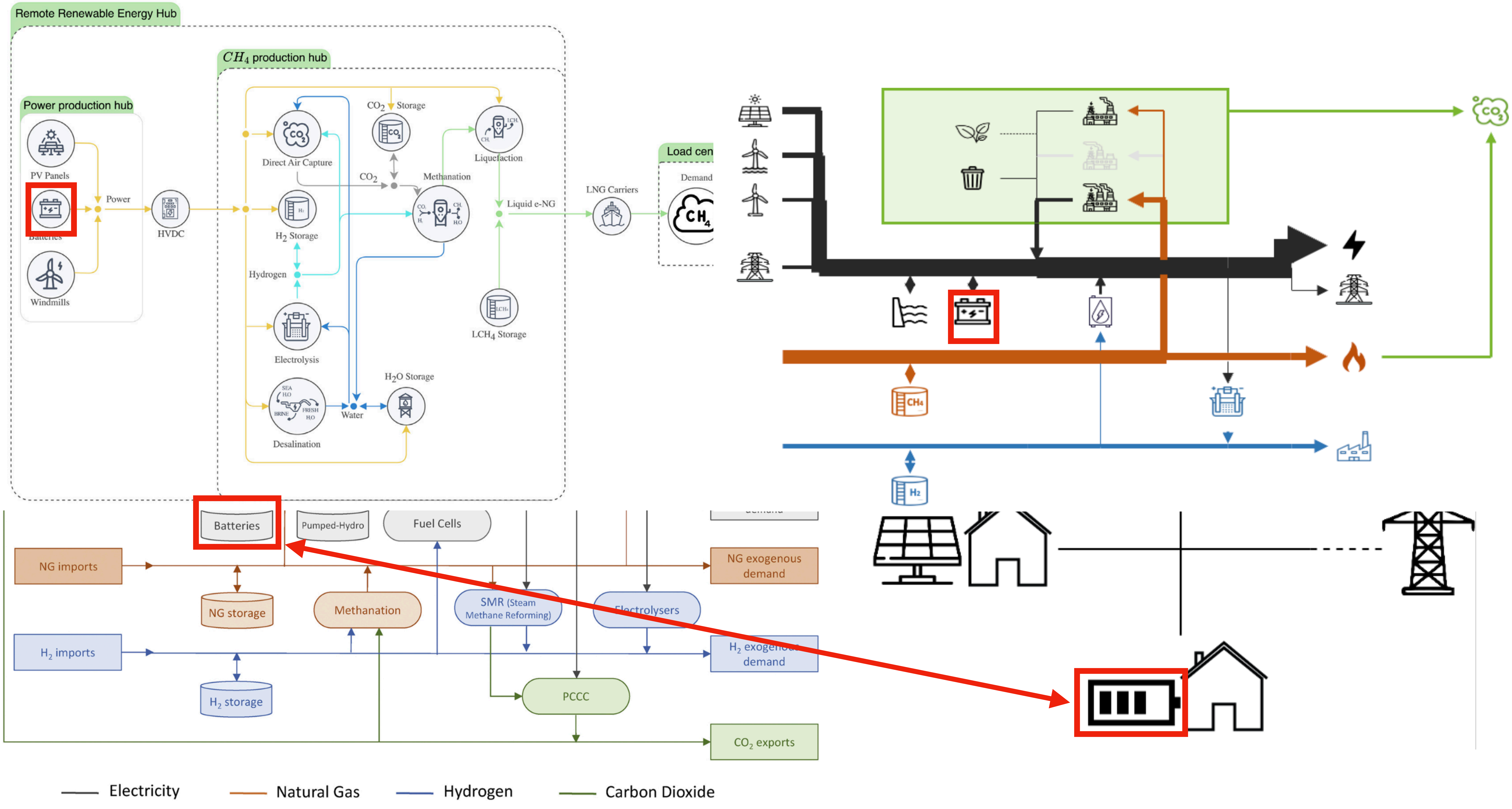
State of the lab

- Working on energy system planning and sizing

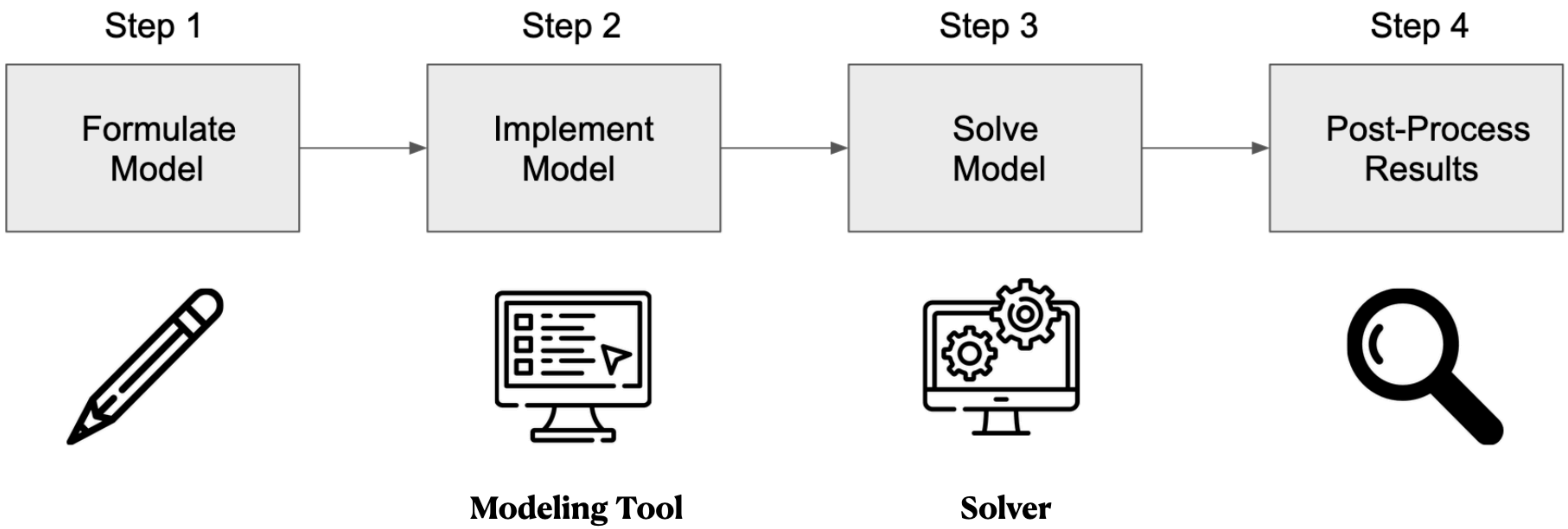


nd sizing


Remote Renewable Energy Hub
CH_4 production hub
Power production hub
PV Panels
Batteries
Windmills
Power
HVDC
Direct Air Capture
CO_2 Storage
CO_2
Methanation
Liquefaction
H_2 Storage
Hydrogen
Electrolysis
Desalination
Water
H_2O Storage
Liquid e-NG
LCH_4 Storage
LNG Carriers
Load center
Demand
CH_4
Solar
fshore Wind
nshore Wind
ricity exogenous demand
ctrical vehicles demand
Batteries
Pumped-Hydro
Fuel Cells
NG imports
NG storage
Methanation
SMR (Steam Methane Reforming)
Electrolysers
NG exogenous demand
H_2 imports
H_2 storage
H_2 exogenous demand
PCCC
CO_2 exports
Electricity
Natural Gas
Hydrogen
Carbon Dioxide


Remote Renewable Energy Hub
CH_4 production hub
Power production hub
PV Panels
Batteries
Windmills
Power
HVDC
Direct Air Capture
CO_2 Storage
CO_2
Liquefaction
Methanation
H_2 Storage
Hydrogen
Liquid e-NG
LNG Carriers
Load cen
Demand
CH_4
LCH_4 Storage
Electrolysis
H_2O Storage
Water
Desalination
CO_2
CH_4
H_2
Batteries
Pumped-Hydro
Fuel Cells
NG imports
NG exogenous demand
NG storage
Methanation
SMR (Steam Methane Reforming)
Electrolysers
H_2 imports
H_2 exogenous demand
H_2 storage
PCCC
CO_2 exports
Electricity
Natural Gas
Hydrogen
Carbon Dioxide

Modeling workflow



Step 1

Formulate Model

Step 2

Implement Model

Step 3

Solve Model

Step 4

Post-Process Results

Modeling Tool

Solver

Go back to 2021

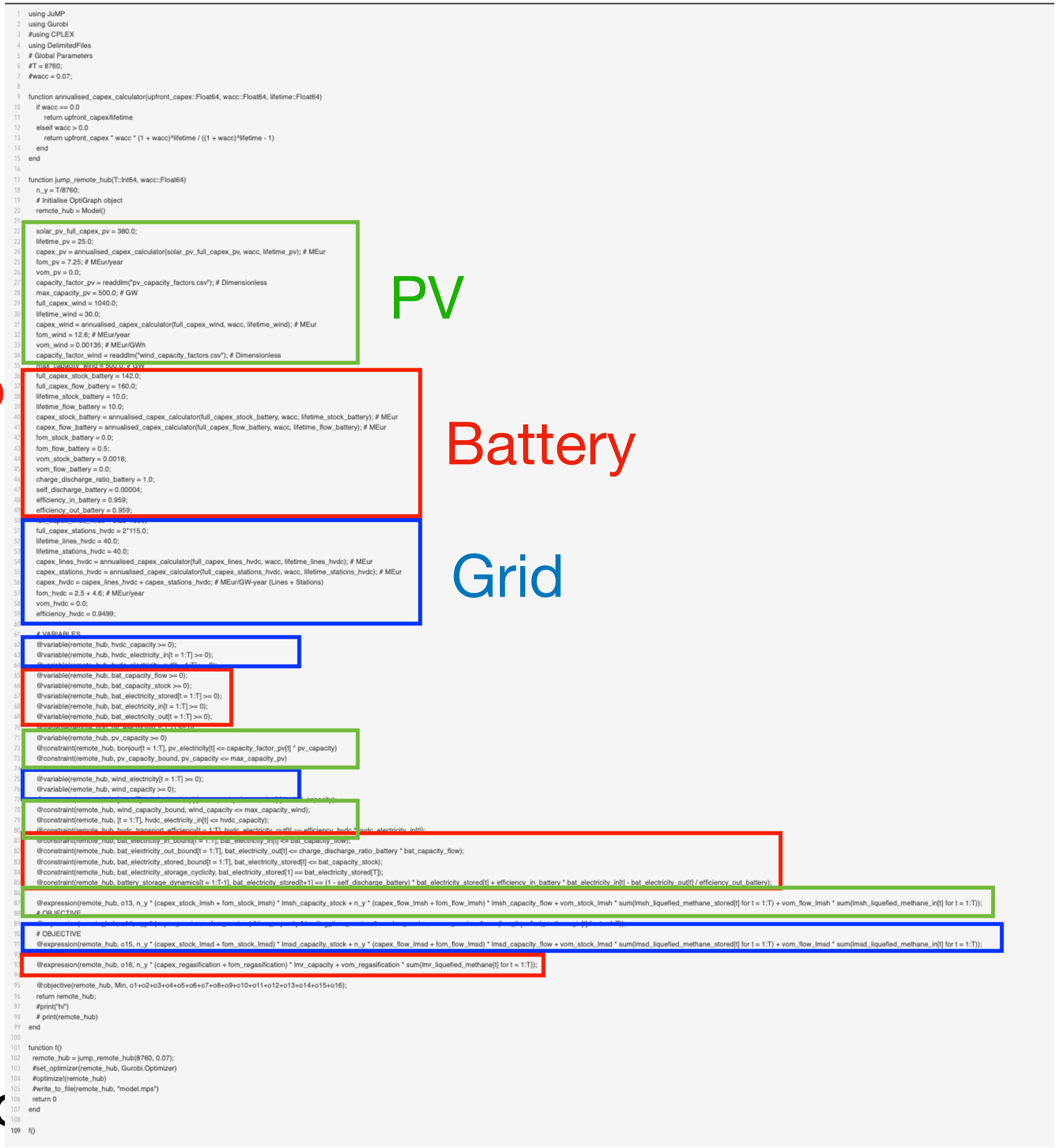
Issues

- Pyomo, JuMP & GAMS were all in use
    - No consensus on the modeling tool to use
- When the researchers left …
    - Difficult to reuse the models
    - Loss of knowledge and expertise
- When a researcher integrated the group …
    - Had to start from a blank sheet
- Little synergies in terms of combining models

Go back to 2021

Issues

- Pyomo, JuMP & GAMS were all in use
  - No consensus on the modeling tool to
- When the researchers left …
  - Difficult to reuse the models
  - Loss of knowledge and expertise
- When a researcher integrated the group
  - Had to start from a blank sheet
- Little synergies in terms of combining mo



```
using JuMP
using Gurobi
#using CPLEX
using DelimitedFiles
# Global Parameters
#T = 8760;
#wacc = 0.07;

function annualised_capex_calculator(upfront_capex::Float64, wacc::Float64, lifetime::Float64)
   if wacc == 0.0
      return upfront_capex/lifetime
   elseif wacc > 0.0
      return upfront_capex * wacc * (1 + wacc)^lifetime / ((1 + wacc)^lifetime - 1)
   end
end

function jump_remote_hub(T::Int64, wacc::Float64)
   n_y = T/8760;
   # Initialise OptiGraph object
   remote_hub = Model()

   solar_pv_full_capex_pv = 380.0;
   lifetime_pv = 25.0;
   capex_pv = annualised_capex_calculator(solar_pv_full_capex_pv, wacc, lifetime_pv); # MEur
   fom_pv = 7.25; # MEur/year
   vom_pv = 0.0;
   capacity_factor_pv = readdlm("pv_capacity_factors.csv"); # Dimensionless
   max_capacity_pv = 500.0; # GW
   full_capex_wind = 1040.0;
   lifetime_wind = 30.0;
   capex_wind = annualised_capex_calculator(full_capex_wind, wacc, lifetime_wind); # MEur
   fom_wind = 12.6; # MEur/year
   vom_wind = 0.00135; # MEur/GWh
   capacity_factor_wind = readdlm("wind_capacity_factors.csv"); # Dimensionless
   max_capacity_wind = 500.0; # GW
   full_capex_stock_battery = 142.0;
   full_capex_flow_battery = 160.0;
   lifetime_stock_battery = 10.0;
   lifetime_flow_battery = 10.0;
   capex_stock_battery = annualised_capex_calculator(full_capex_stock_battery, wacc, lifetime_stock_battery); # MEur
   capex_flow_battery = annualised_capex_calculator(full_capex_flow_battery, wacc, lifetime_flow_battery); # MEur
   fom_stock_battery = 0.0;
   fom_flow_battery = 0.5;
   vom_stock_battery = 0.0018;
   vom_flow_battery = 0.0;
   charge_discharge_ratio_battery = 1.0;
   self_discharge_battery = 0.00004;
   efficiency_in_battery = 0.959;
   efficiency_out_battery = 0.959;
   [illegible]
   full_capex_stations_hvdc = 2*115.0;
   lifetime_lines_hvdc = 40.0;
   lifetime_stations_hvdc = 40.0;
   capex_lines_hvdc = annualised_capex_calculator(full_capex_lines_hvdc, wacc, lifetime_lines_hvdc); # MEur
   capex_stations_hvdc = annualised_capex_calculator(full_capex_stations_hvdc, wacc, lifetime_stations_hvdc); # MEur
   capex_hvdc = capex_lines_hvdc + capex_stations_hvdc; # MEur/GW-year (Lines + Stations)
   fom_hvdc = 2.5 + 4.6; # MEur/year
   vom_hvdc = 0.0;
   efficiency_hvdc = 0.9499;

   # VARIABLES
   @variable(remote_hub, hvdc_capacity >= 0);
   @variable(remote_hub, hvdc_electricity_in[t = 1:T] >= 0);
   @variable(remote_hub, hvdc_electricity_out[t = 1:T] >= 0);
   @variable(remote_hub, bat_capacity_flow >= 0);
   @variable(remote_hub, bat_capacity_stock >= 0);
   @variable(remote_hub, bat_electricity_stored[t = 1:T] >= 0);
   @variable(remote_hub, bat_electricity_in[t = 1:T] >= 0);
   @variable(remote_hub, bat_electricity_out[t = 1:T] >= 0);
   @variable(remote_hub, pv_electricity[t = 1:T] >= 0)
   @variable(remote_hub, pv_capacity >= 0)
   @constraint(remote_hub, bonjour[t = 1:T], pv_electricity[t] <= capacity_factor_pv[t] * pv_capacity)
   @constraint(remote_hub, pv_capacity_bound, pv_capacity <= max_capacity_pv)

   @variable(remote_hub, wind_electricity[t = 1:T] >= 0);
   @variable(remote_hub, wind_capacity >= 0);
   [illegible]
   @constraint(remote_hub, wind_capacity_bound, wind_capacity <= max_capacity_wind);
   @constraint(remote_hub, [t = 1:T], hvdc_electricity_in[t] <= hvdc_capacity);
   @constraint(remote_hub, hvdc_transport_efficiency[t = 1:T], hvdc_electricity_out[t] == efficiency_hvdc * hvdc_electricity_in[t]);
   @constraint(remote_hub, bat_electricity_in_bound[t = 1:T], bat_electricity_in[t] <= bat_capacity_flow);
   @constraint(remote_hub, bat_electricity_out_bound[t = 1:T], bat_electricity_out[t] <= charge_discharge_ratio_battery * bat_capacity_flow);
   @constraint(remote_hub, bat_electricity_stored_bound[t = 1:T], bat_electricity_stored[t] <= bat_capacity_stock);
   @constraint(remote_hub, bat_electricity_storage_cyclicity, bat_electricity_stored[1] == bat_electricity_stored[T]);
   @constraint(remote_hub, battery_storage_dynamics[t = 1:T-1], bat_electricity_stored[t+1] == (1 - self_discharge_battery) * bat_electricity_stored[t] + efficiency_in_battery * bat_electricity_in[t] - bat_electricity_out[t] / efficiency_out_battery);

   @expression(remote_hub, o13, n_y * (capex_stock_lmsh + fom_stock_lmsh) * lmsh_capacity_stock + n_y * (capex_flow_lmsh + fom_flow_lmsh) * lmsh_capacity_flow + vom_stock_lmsh * sum(lmsh_liquefied_methane_stored[t] for t = 1:T) + vom_flow_lmsh * sum(lmsh_liquefied_methane_in[t] for t = 1:T));
   # OBJECTIVE
   [illegible]
   # OBJECTIVE
   @expression(remote_hub, o15, n_y * (capex_stock_lmsd + fom_stock_lmsd) * lmsd_capacity_stock + n_y * (capex_flow_lmsd + fom_flow_lmsd) * lmsd_capacity_flow + vom_stock_lmsd * sum(lmsd_liquefied_methane_stored[t] for t = 1:T) + vom_flow_lmsd * sum(lmsd_liquefied_methane_in[t] for t = 1:T));

   @expression(remote_hub, o16, n_y * (capex_regasification + fom_regasification) * lmr_capacity + vom_regasification * sum(lmr_liquefied_methane[t] for t = 1:T));

   @objective(remote_hub, Min, o1+o2+o3+o4+o5+o6+o7+o8+o9+o10+o11+o12+o13+o14+o15+o16);
   return remote_hub;
   #print("hi")
   # print(remote_hub)
end

function f()
  remote_hub = jump_remote_hub(8760, 0.07);
  #set_optimizer(remote_hub, Gurobi.Optimizer)
  #optimize!(remote_hub)
  #write_to_file(remote_hub, "model.mps")
  return 0
end

f()
```

Go back to 2021

Task

Uniformisation of the modeling tool in use

Go back to 2021

Task

Uniformisation of the modeling tool in use

So our journey begins

Go back to 2021

Task

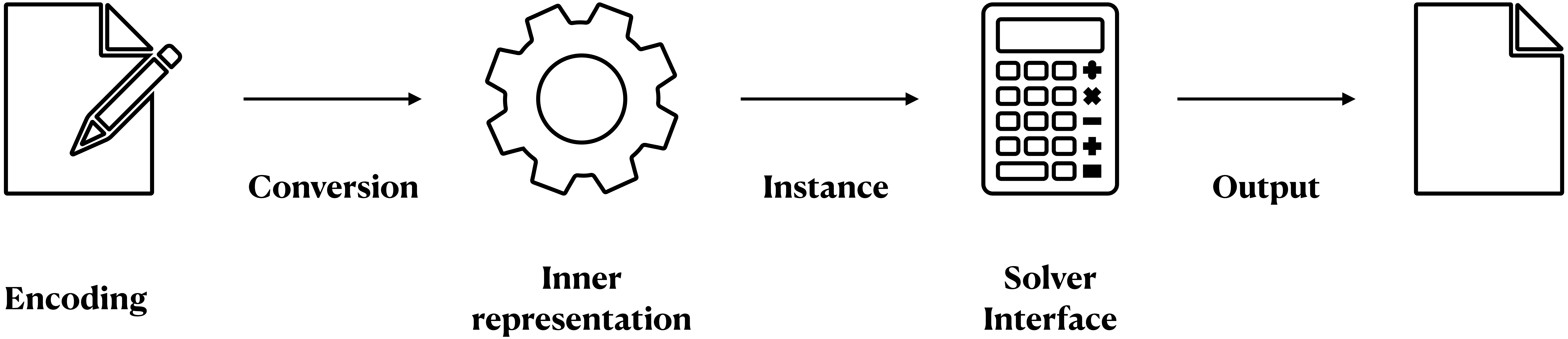
Uniformisation of the modeling tool in use

So our journey begins

(And I got hired 😅)

Modeling tools 1.0.1

Inner workings of modeling tools



Finding a modeling tool

Step 1: Know your community

Researchers in the lab

- A lot had background in energy (little programming knowledge)
  - Core users

- A few had background in computer science and optimization
  - Advanced users

Going beyond our lab

Finding a modeling tool

Step 2: List your requirements

- Mixed Integer Linear Programming
- Stand-alone and lightweight (to be in python)
- Model reuse and modular construction
- Structured models
- Multiple solvers
- Open-source

Finding a modeling tool

Step 3: Find a good fit

- **A**lgebraic **m**odeling **l**anguages (AMLs):
  - Formulation close to the mathematical one
  - Very expressive
  - Interfaces with multiple solvers
  - Do not expect structure

GAMS

AMPL
STREAMLINED MODELING
FOR REAL OPTIMIZATION

PYOMO

JUMP

Finding a modeling tool

Step 3: Find a good fit

- **A**lgebraic **m**odeling **l**anguages (AMLs):
  - Formulation close to the mathematical one
  - Very expressive
  - Interfaces with multiple solvers
  - Do not expect structure

GAMS

JUMP

AMPL
STREAMLINED MODELING FOR REAL OPTIMIZATION

PYOMO

- **O**bject-**o**riented **m**odeling **e**nvironment (OOMEs):
  - Application focused
  - A finite set of predefined components that can be reused
  - Difficult to add/modify components
  - Tailored analysis tools

PyPSA

PLEXOS

Calliope

BALMOREL
Energy system model

Finding a modeling tool

Tool found

Job done congrats ! 🥳

Our requirements

Step 4: Find your niche

Requirements	Algebraic modeling languages	Object-oriented modeling environments
MILP	✅	❌
Standalone & lightweight	🟠	❌
Modular & reuse	❌	🟠
Structured models	🟠	✅
Multiple solvers	✅	✅
Open-source	🟠	🟠

Our requirements

Step 4: Find your niche

Gap ↔

Requirements	Algebraic modeling languages	Object-oriented modeling environments
MILP	✅	❌
Standalone & lightweight	🟠	❌
Modular & reuse	❌	🟠
Structured models	🟠	✅
Multiple solvers	✅	✅
Open-source	🟠	🟠

Creating a modeling tool

Step 5: Make the tool

- The Graph-Based Optimization Modeling Language (GBOML)
    - Encoding by writing equations
    - Close to the Algebraic Modeling Languages
    - Can encode any MILP

Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅					

Creating a modeling tool

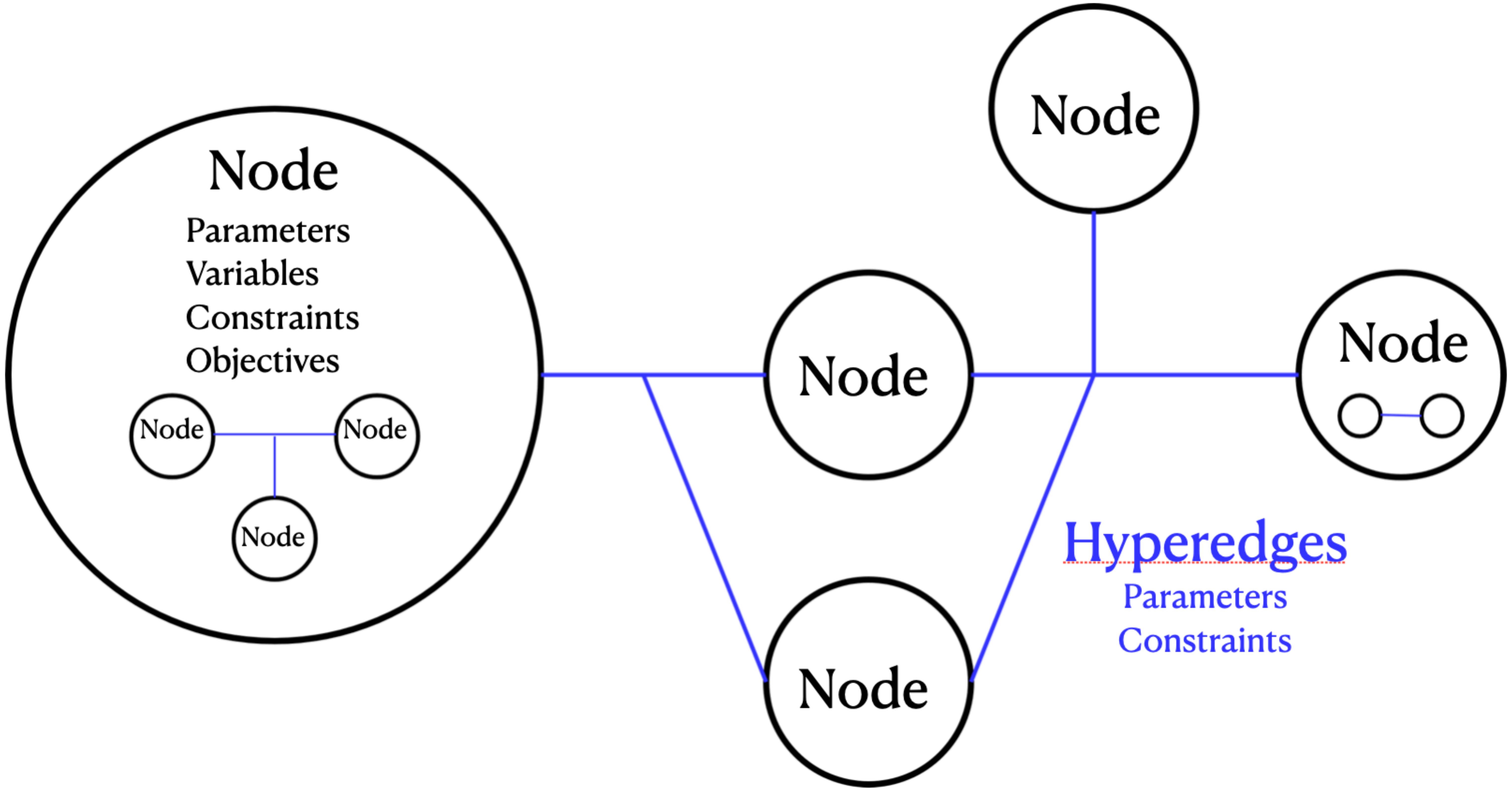
Step 5: Make the tool

- The Graph-Based Optimization Modeling Language (GBOML)
  - Coded in Python
  - Very few dependencies
  - Easy to install and deploy

Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅				

Creating a modeling tool

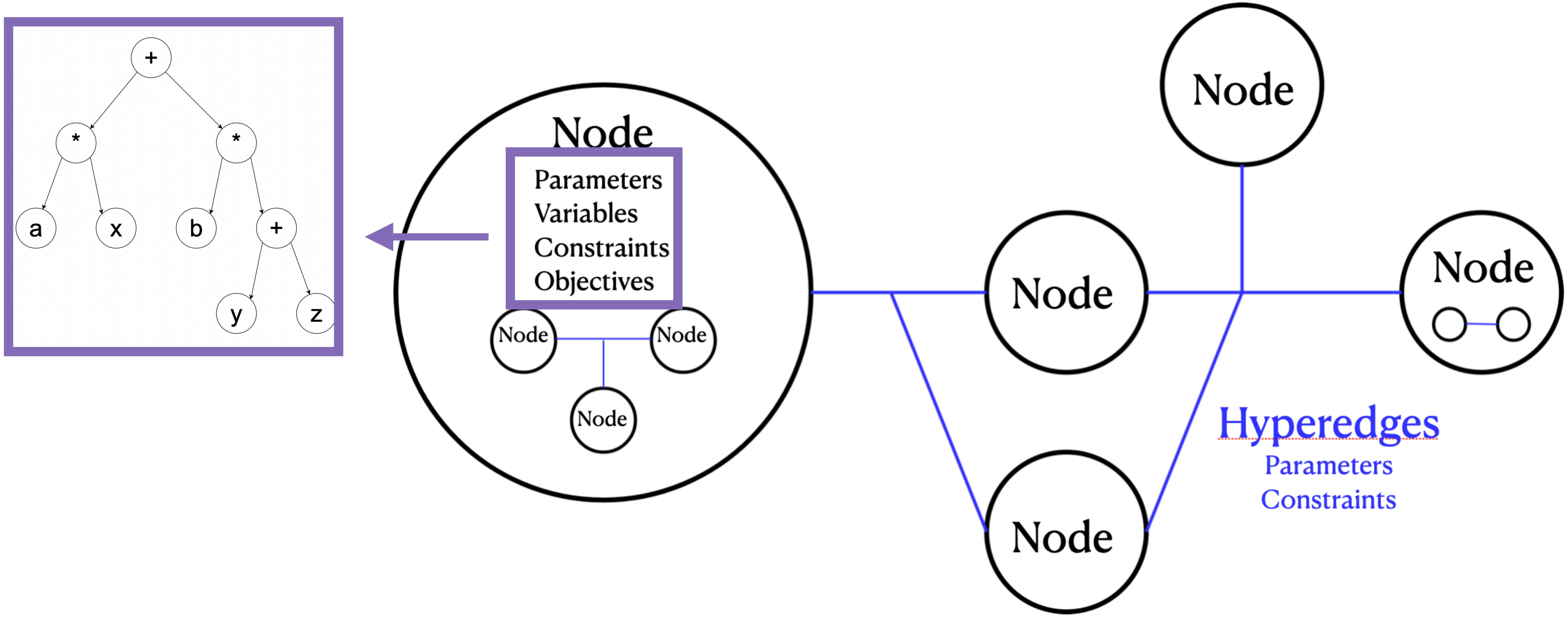
Step 5: Make the tool



Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅		

Creating a modeling tool

Step 5: Make the tool



Node

Parameters
Variables
Constraints
Objectives

Node

Node

Node

Node

Node

Node

Node

Node

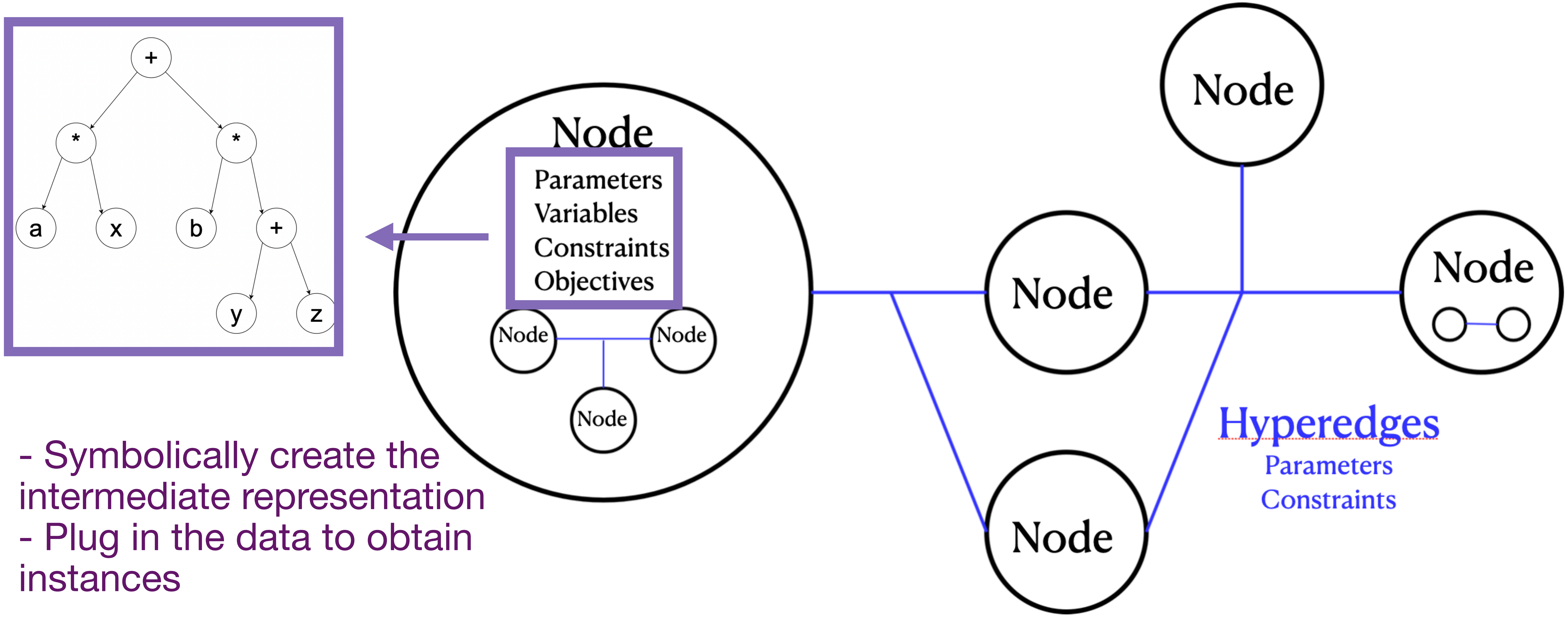
Hyperedges

Parameters
Constraints

Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅		

Creating a modeling tool

Step 5: Make the tool

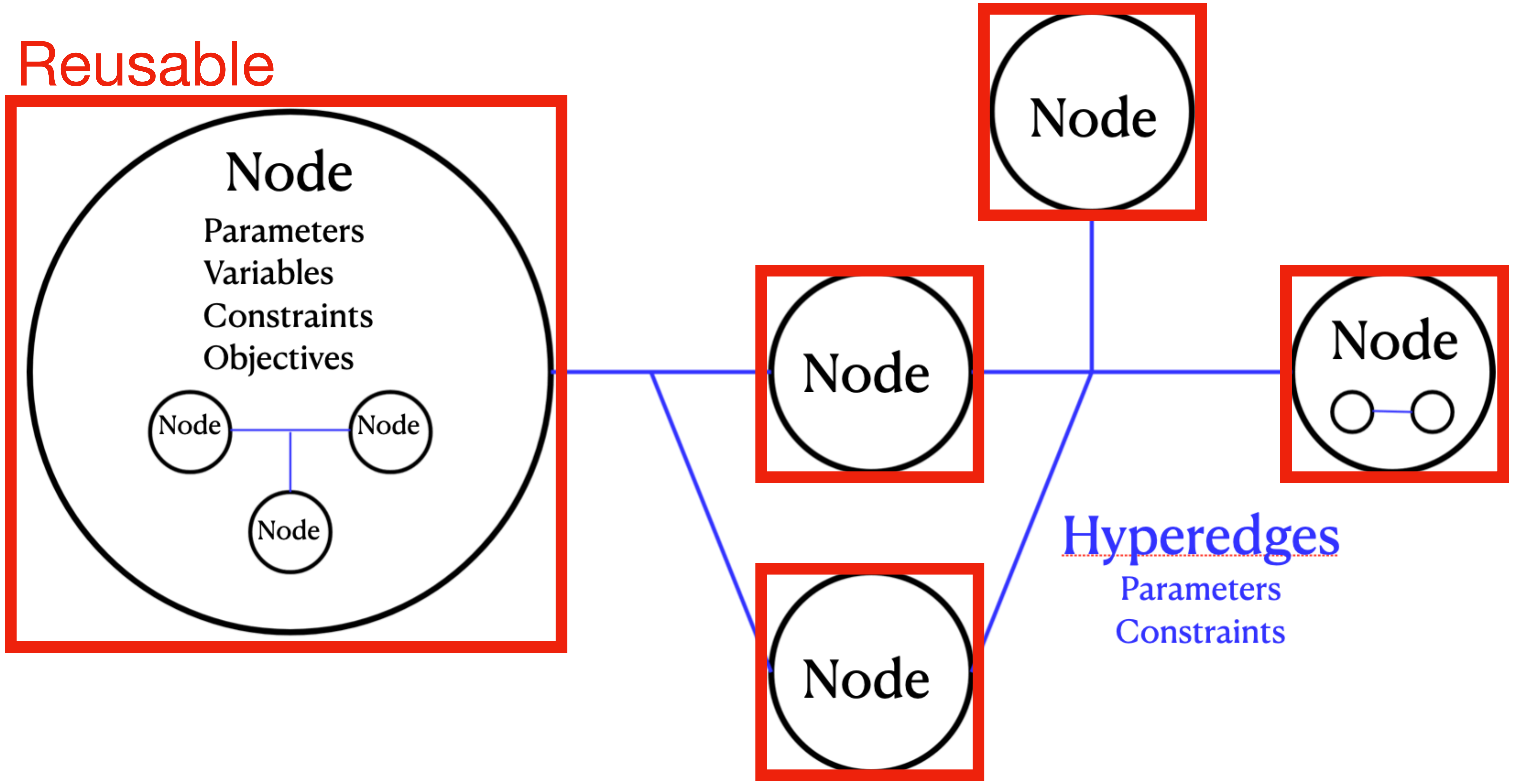


- Symbolically create the intermediate representation
- Plug in the data to obtain instances

Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅		

Creating a modeling tool

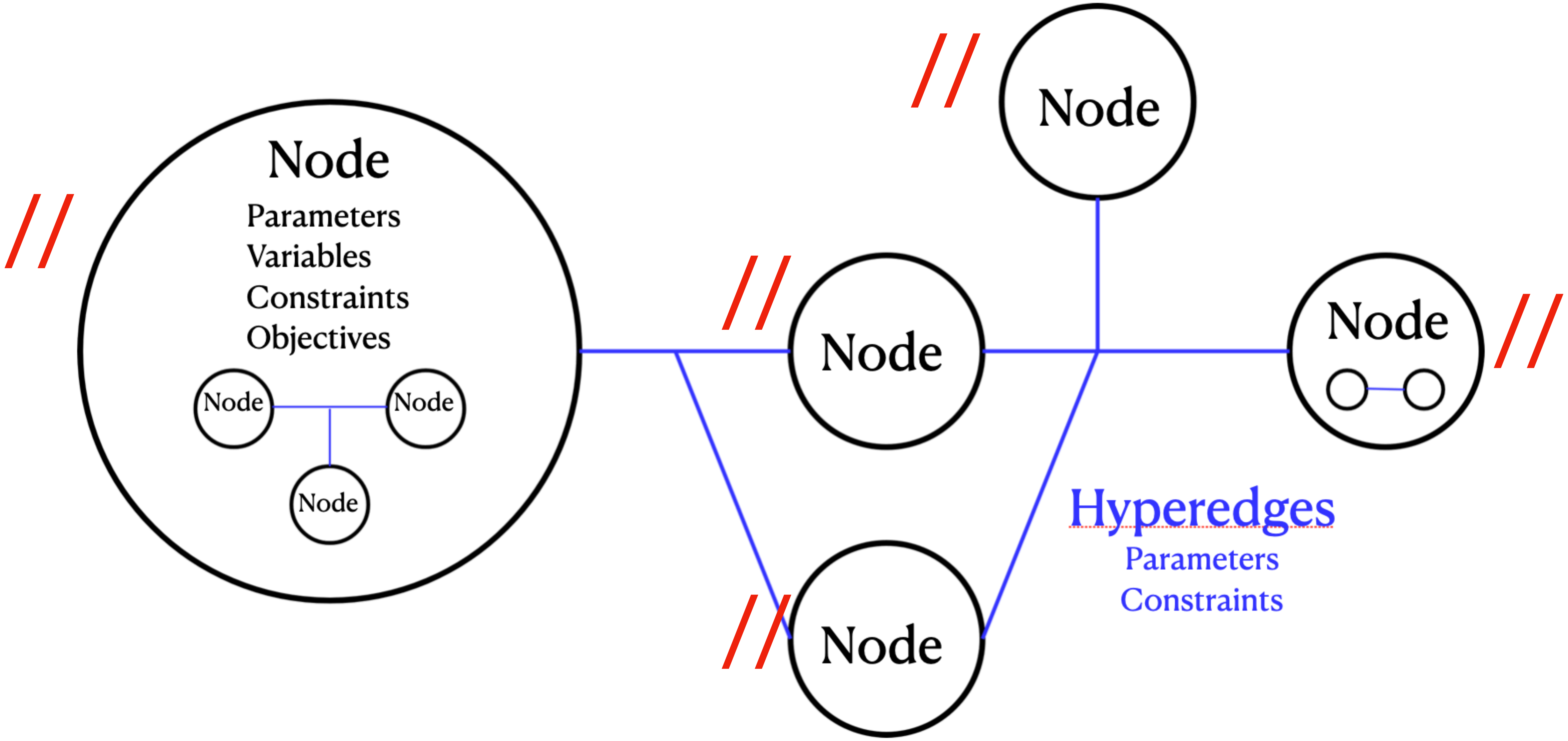
Step 5: Make the tool



Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅		

Creating a modeling tool

Step 5: Make the tool



Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅		

Creating a modeling tool

Step 5: Make the tool

```
#NODE <node_name>
#PARAMETERS
<parameter_def>
#VARIABLES
<variable_def>
#CONSTRAINTS
<constraint_def>
#OBJECTIVES
<objective_def>
```

```
#HYPEREDGE <edge_name>
#PARAMETERS
<parameter_def>
#CONSTRAINTS
<constraint_def>
```

Creating a modeling tool

Step 5: Make the tool

#NODE <node_name>
#PARAMETERS
<parameter_def>
#VARIABLES
<variable_def>
#CONSTRAINTS
<constraint_def>
#OBJECTIVES
<objective_def>

#HYPEREDGE <edge_name>
#PARAMETERS
<parameter_def>
#CONSTRAINTS
<constraint_def>

Keep things as simple as possible

Creating a modeling tool

Step 5: Make the tool

- The Graph-Based Optimization Modeling Language (GBOML)
  - Interfaces with

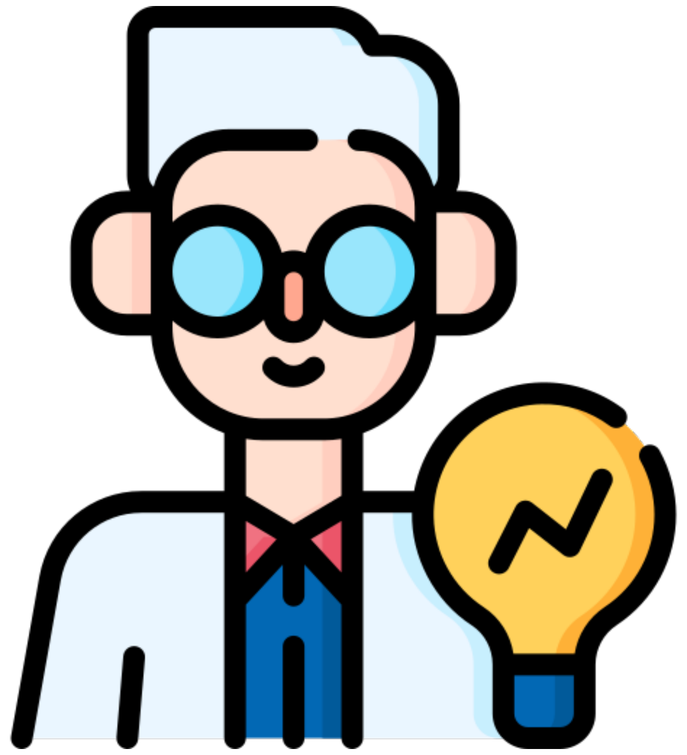
HiGHS GUROBI OPTIMIZATION FICO™ IBM COIN|OR COMPUTATIONAL INFRASTRUCTURE FOR OPERATIONS RESEARCH

- Structure exploiting methods
  - DSP[19]: Dantzig-Wolfe decomposition
  - CPLEX: Benders decomposition
- Released under MIT license

Requirements	MILP	Standalone & lightweight	Modular & reuse	Structured models	Multiple solvers	Open source
GBOML	✅	✅	✅	✅	✅	✅

Creating a modeling tool

Structural change in the lab

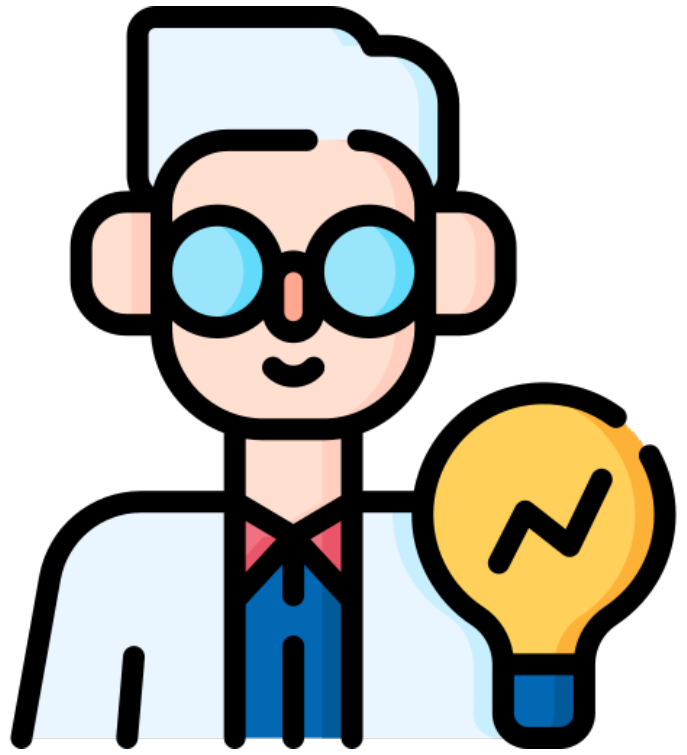


- Expert users
  - Create a library of nodes
  - Help with debugging
- Core users
  - Tune the nodes
  - Create models by combining nodes, hyperedges and models

Creating a modeling tool

Structural change in the lab

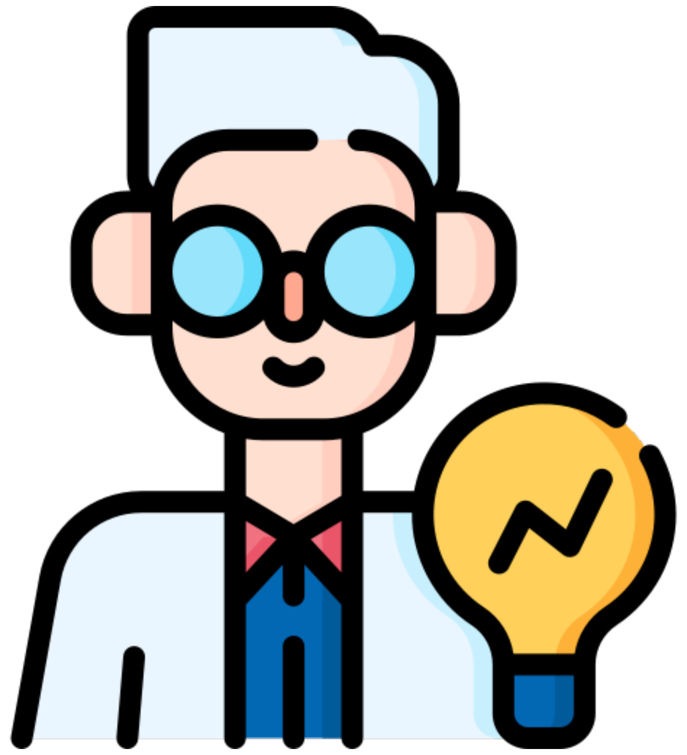
- Expert users
- Generic node definition
- No more model and knowledge loss



- Core users
- Fast models implementation
- Implementation of « what-if scenarios »
- More synergies

Creating a modeling tool

Structural change in the lab



- Expert users
- Generic node definition
- No more model and knowledge loss
- Valorization of the models

- Core users & industry
- Fast models implementation
- Implementation of « what-if scenarios »
- More synergies

Creating a modeling language

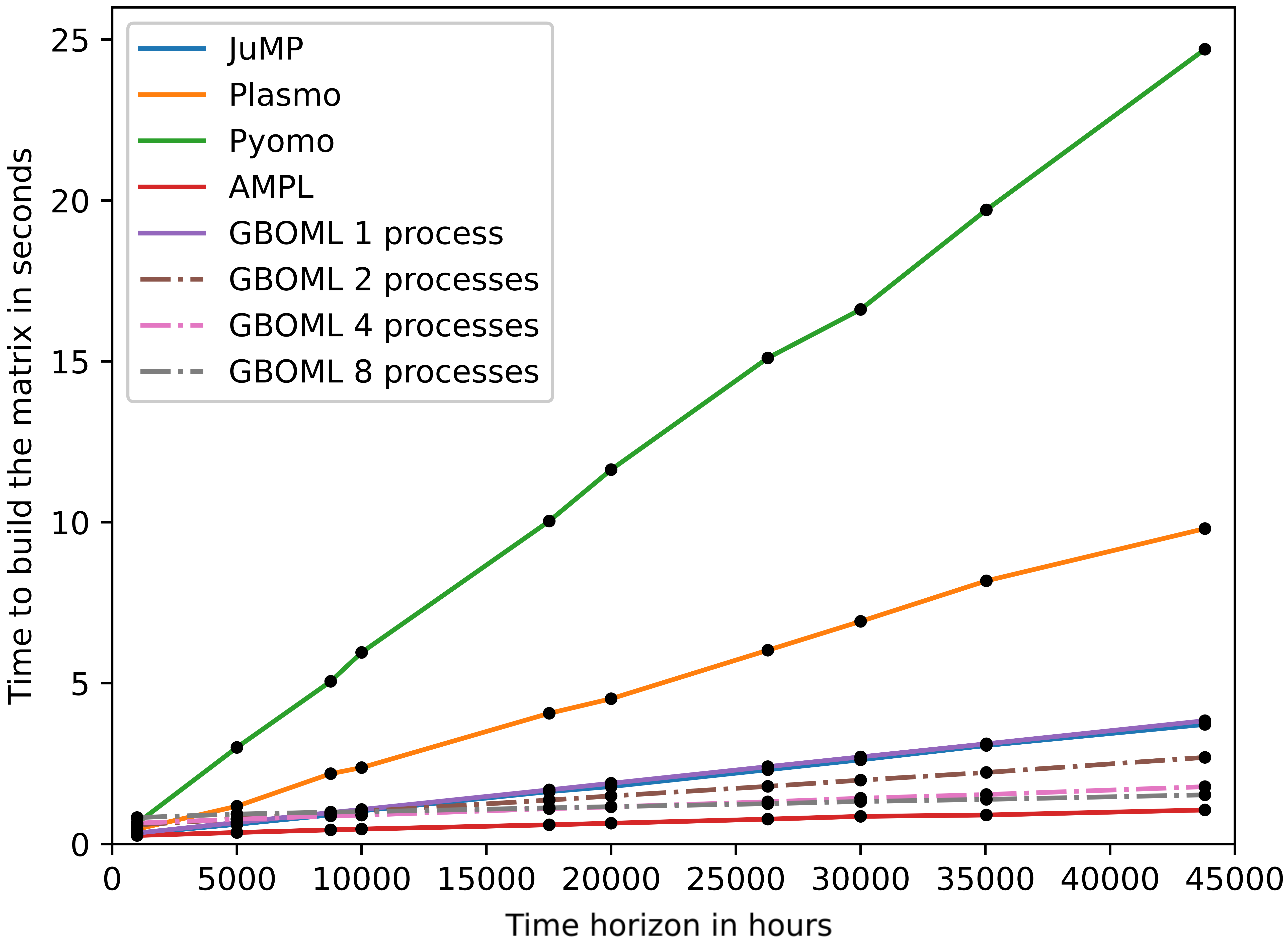
BONUS step: Make your differences count

- Take a big time-dependent model
  - Increase the time horizon
- Compare tools in terms of:
  - Time to build the intermediate representation
  - Peak RAM usage
- Highlight the influence of the structure on the solving time

Creating a modeling language

BONUS step: Make your differences count

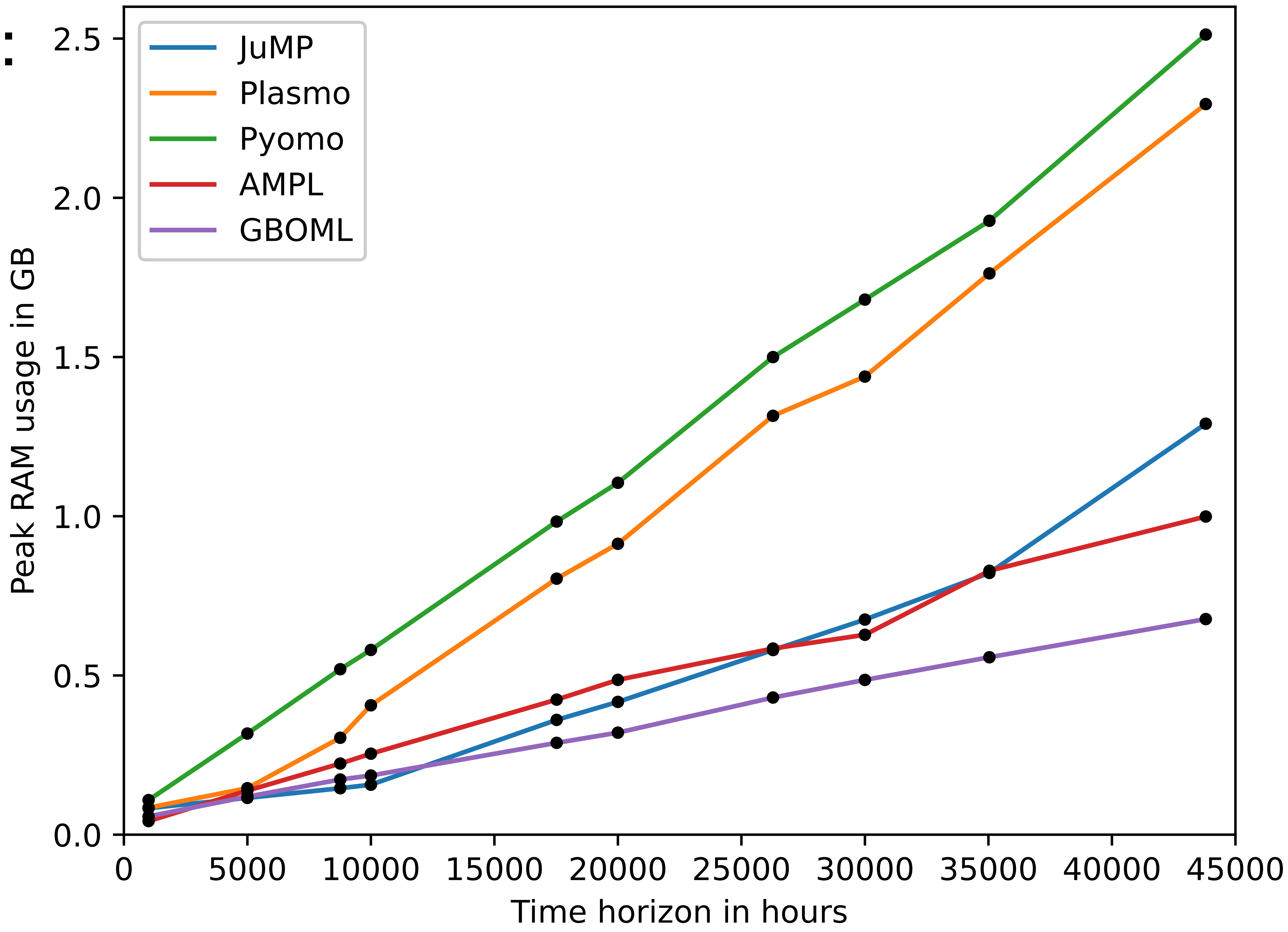
Generation time:



Creating a modeling language

BONUS step: Make your differences count

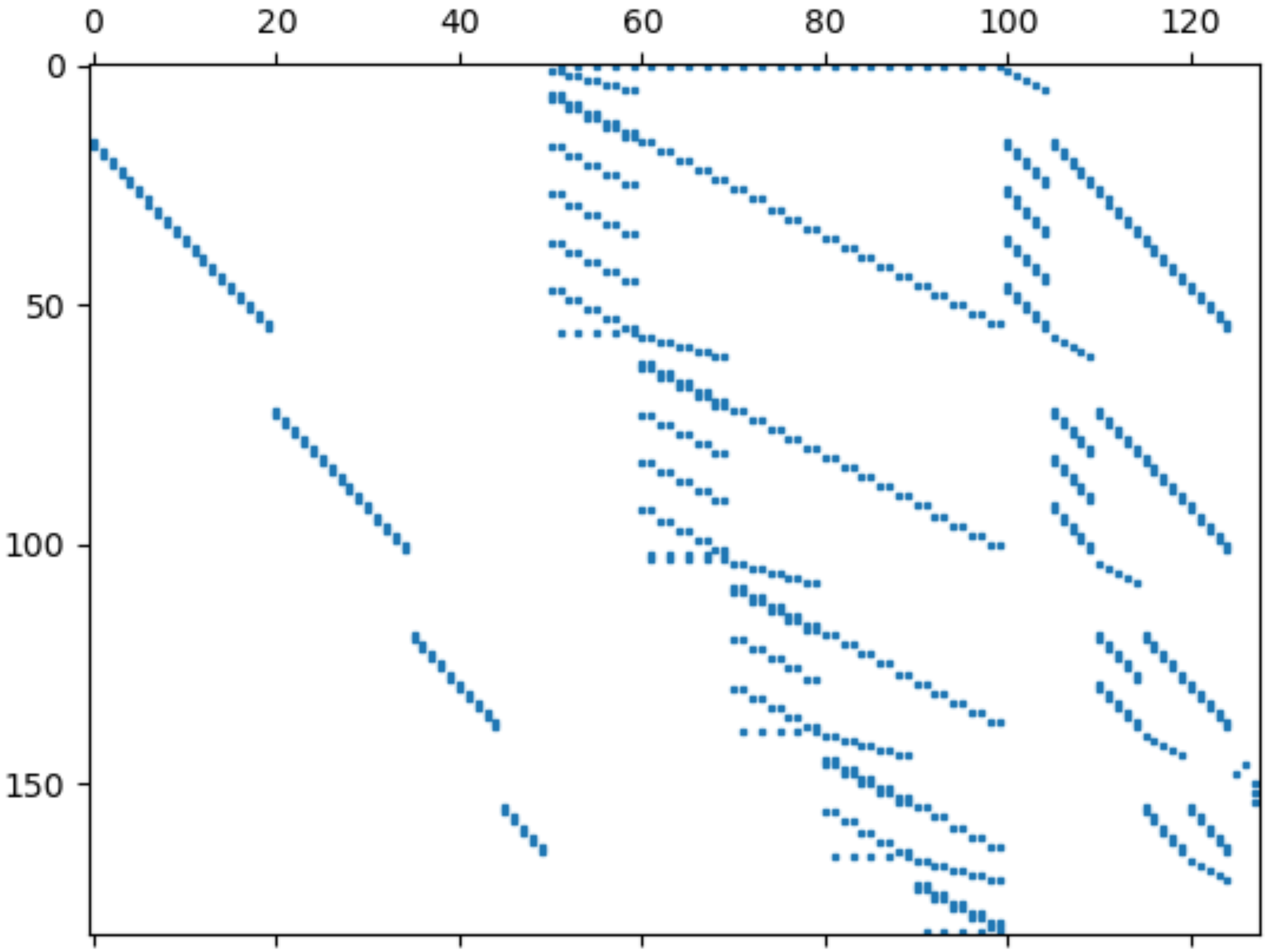
Peak RAM usage:



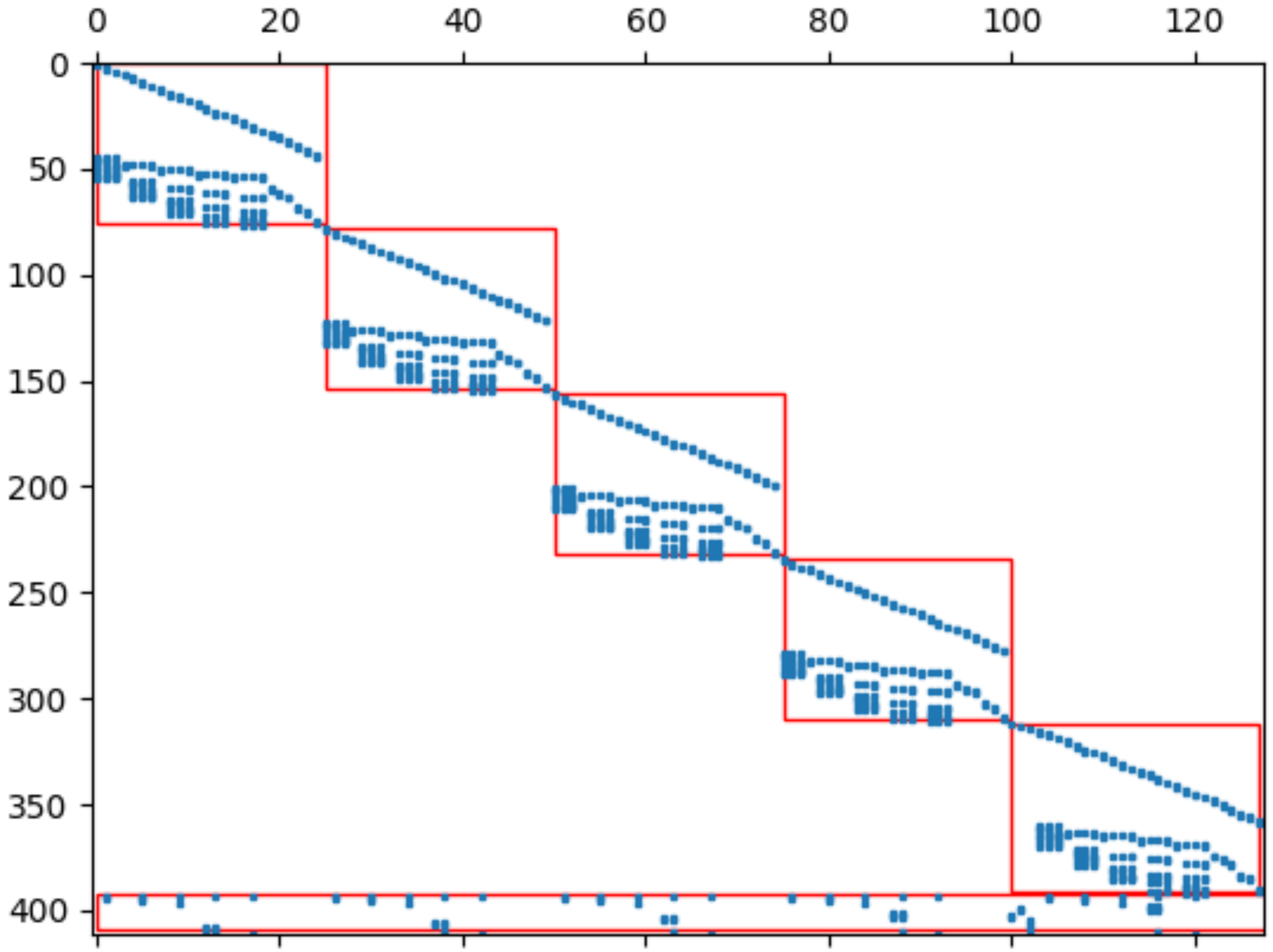
Creating a modeling language

BONUS step: Make your differences count

Solving time:



Solved in 25 seconds by Gurobi



Solved in 2.5 seconds using Dantzig-Wolfe

This talk

Planning

Part 2:

Go beyond state of the art

Going beyond state of the art

Idea 1

- The first modeling tools made linear programming main-stream

> Reliable translation from the «modeler's form» to the «algorithm's form» is often a considerable expense … and error-prone.
>
> - Robert Fourer et al. A MODELING LANGUAGE FOR MATHEMATICAL PROGRAMMING (1990)

Going beyond state of the art

Idea 1

- The first modeling tools made linear programming main-stream
- A lot of methods still have a barrier to entry:
    - Robust optimization reformulation
    - Sensitivity analysis and warm-starting
    - Library modeling

Going beyond state of the art

Idea 1

- The first modeling tools made linear programming main-stream
- A lot of methods still have a barrier to entry:
  - Robust optimization reformulation
  - Sensitivity analysis and warm-starting
  - Library modeling

$\left.\vphantom{\begin{matrix}a\\b\\c\end{matrix}}\right\}$ **Rely on the user**

Going beyond state of the art

Idea 1: Make other methods mainstream

```
#NODE my_node
#PARAMETERS
λ = 35;
b = 12;
#VARIABLES
internal: x;
internal: y;
#CONSTRAINTS
x >= 0;
y >= 0;
λ*x+y >= b;
#OBJECTIVES
min: x+2y
```

Going beyond state of the art

Idea 1: Make other methods mainstream

```
#NODE my_node
#PARAMETERS
λ = 45;
b = 12;
#VARIABLES
internal: x;
internal: y;
#CONSTRAINTS
x >= 0;
y >= 0;
λ*x+y >= b;
#OBJECTIVES
min: x+2y
```

Going beyond state of the art

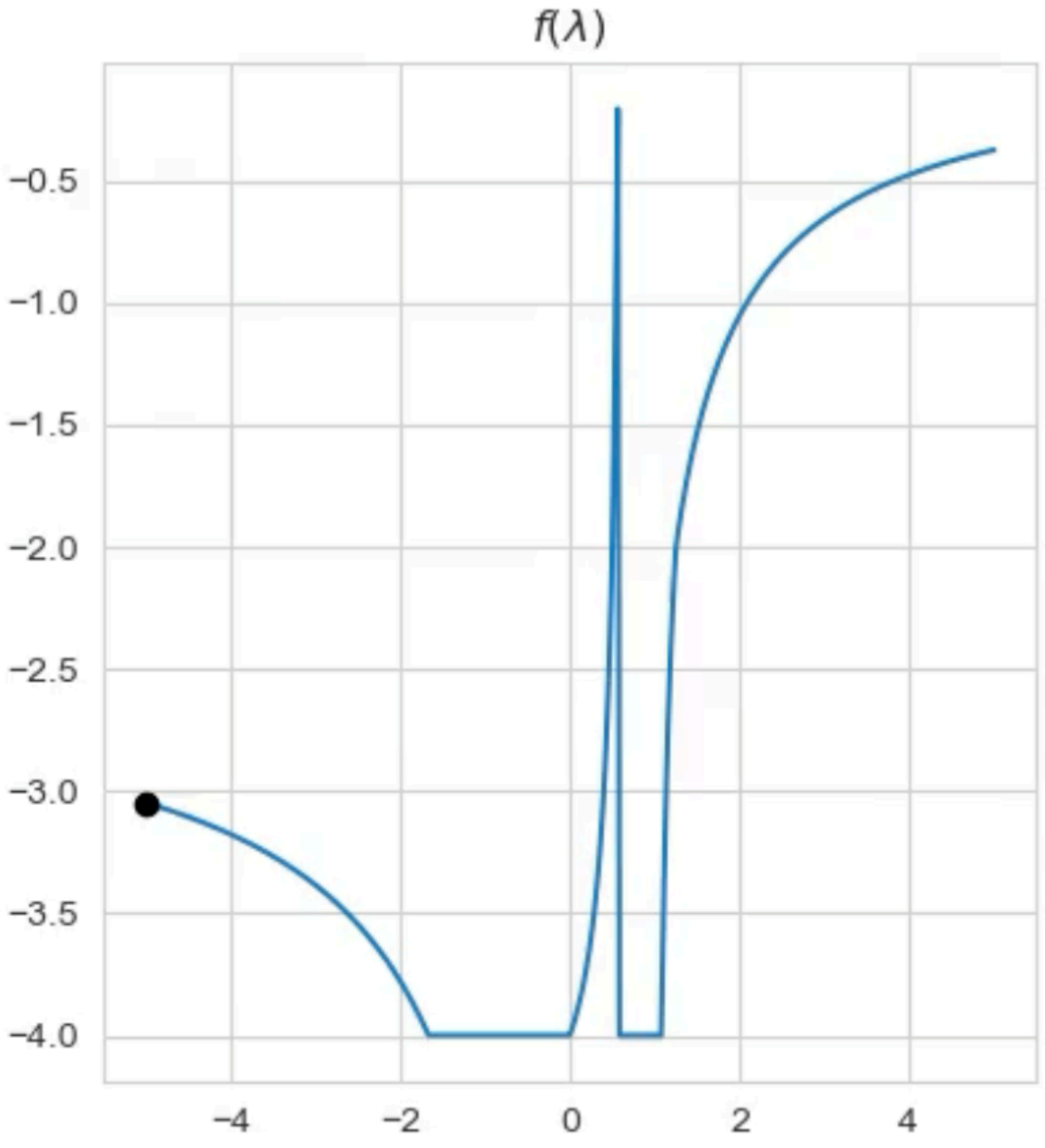
Idea 1: Make other methods mainstream

```
#NODE my_node
#PARAMETERS
λ = 40 + [-5, 5];
b = 12;
#VARIABLES
internal: x;
internal: y;
#CONSTRAINTS
x >= 0;
y >= 0;
λ*x+y >= b;
#OBJECTIVES
min: x+2y
```

Going beyond state of the art

Idea 1: Make other methods mainstream

```
#NODE my_node
#PARAMETERS
λ = 40 + [-5, 5];
b = 12;
#VARIABLES
internal: x;
internal: y;
#CONSTRAINTS
x >= 0;
y >= 0;
λ*x+y >= b;
#OBJECTIVES
min: x+2y
```



Going beyond state of the art

Idea 1: Make other methods mainstream

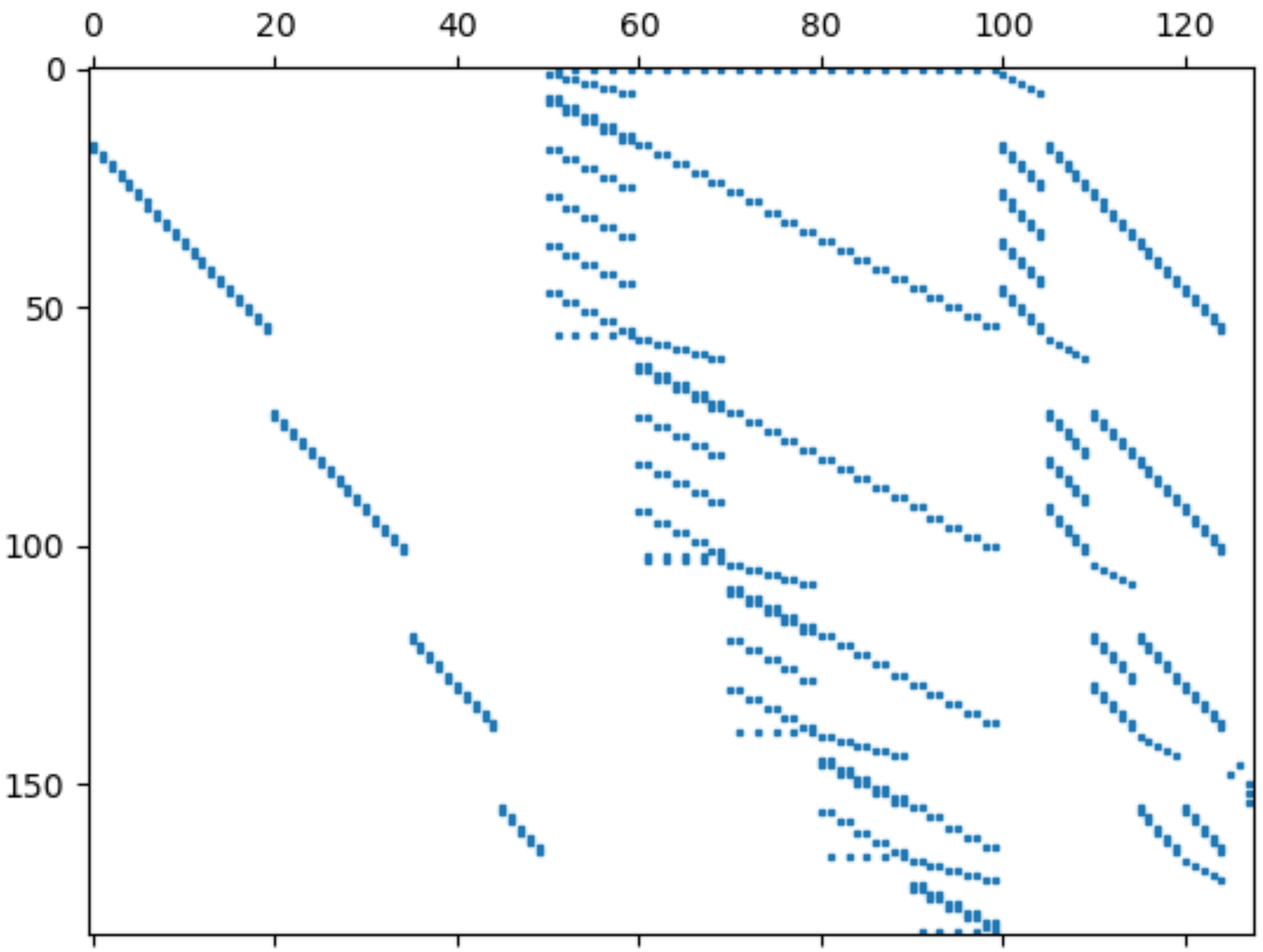
```
#NODE my_node
#PARAMETERS
λ = 40 + [-5, 5];
b = 12;
#VARIABLES
internal: x;
internal: y;
#CONSTRAINTS
x >= 0;
y >= 0;
λ*x+y >= b;
#OBJECTIVES
min: x+2y
```

→ Direct reformulation to robust optimization or sensitivity analysis

Going beyond state of the art

Idea 2: Using syntactic formulation for presolve

Solvers presolve

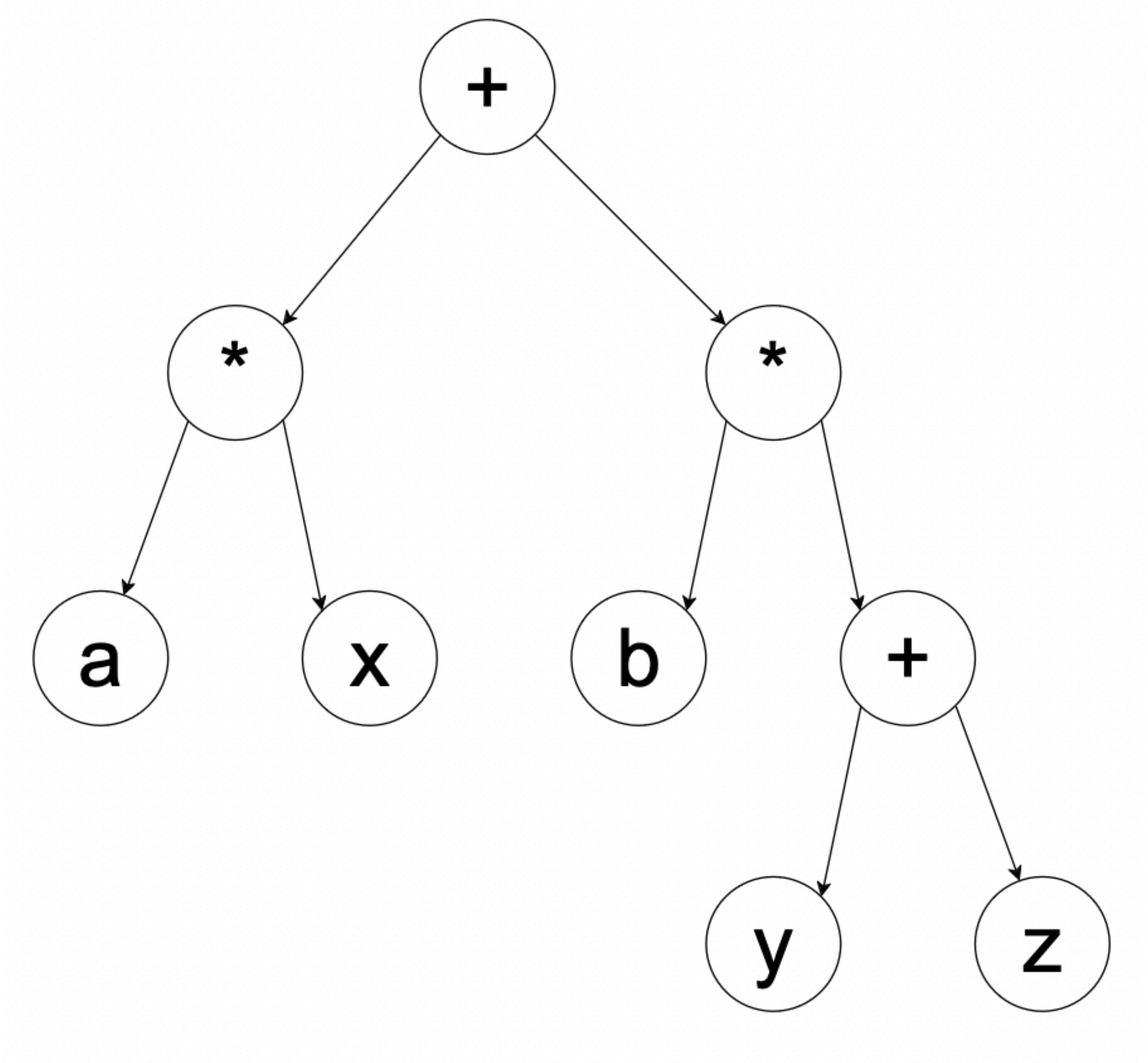


- Try to find structure to reformulate the model
    - Remove constraints & variables
- Lead to problem that can be solved more easily

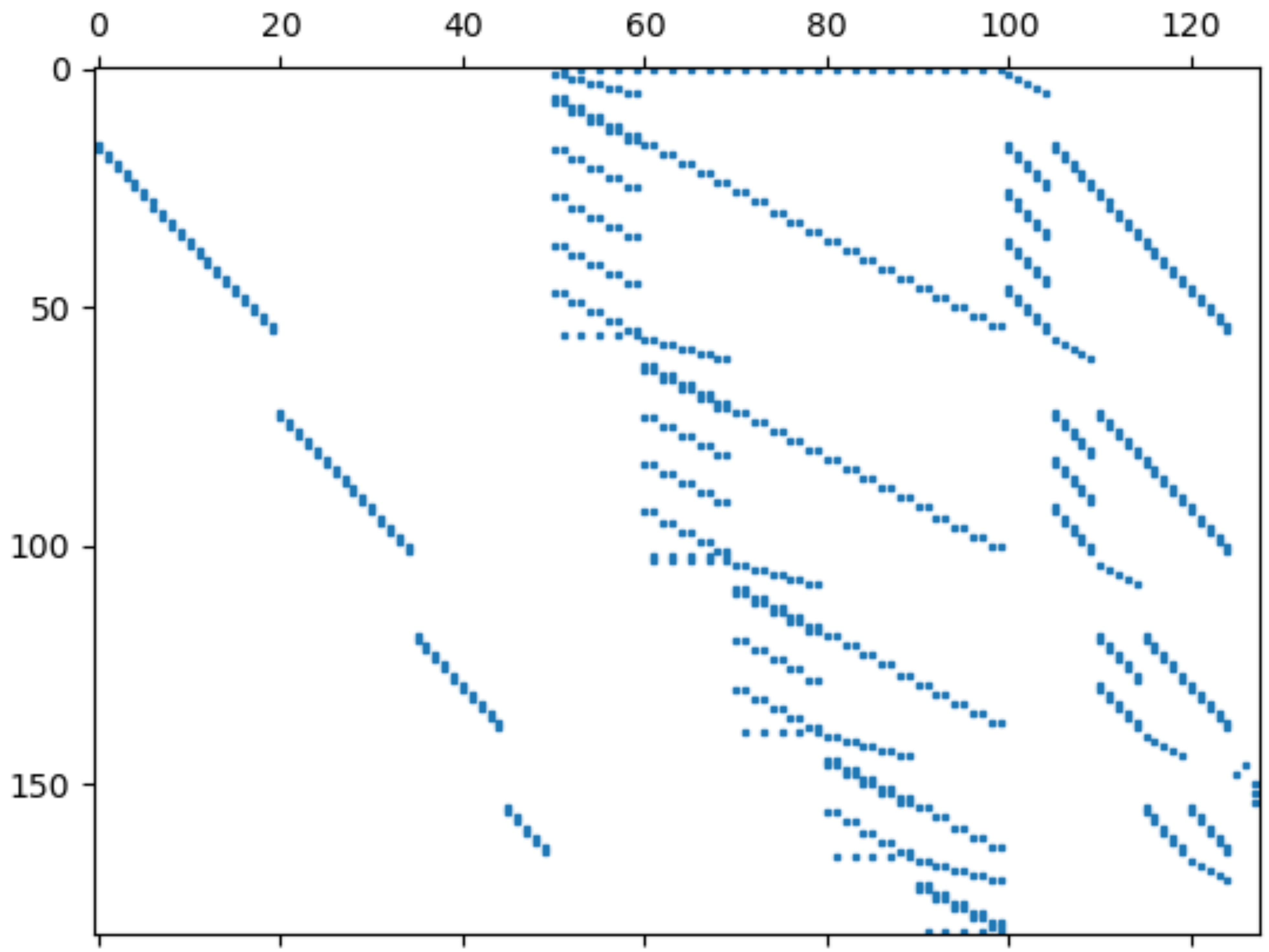
Going beyond state of the art

Idea 2: Using syntactic formulation for presolve

Modeling tools



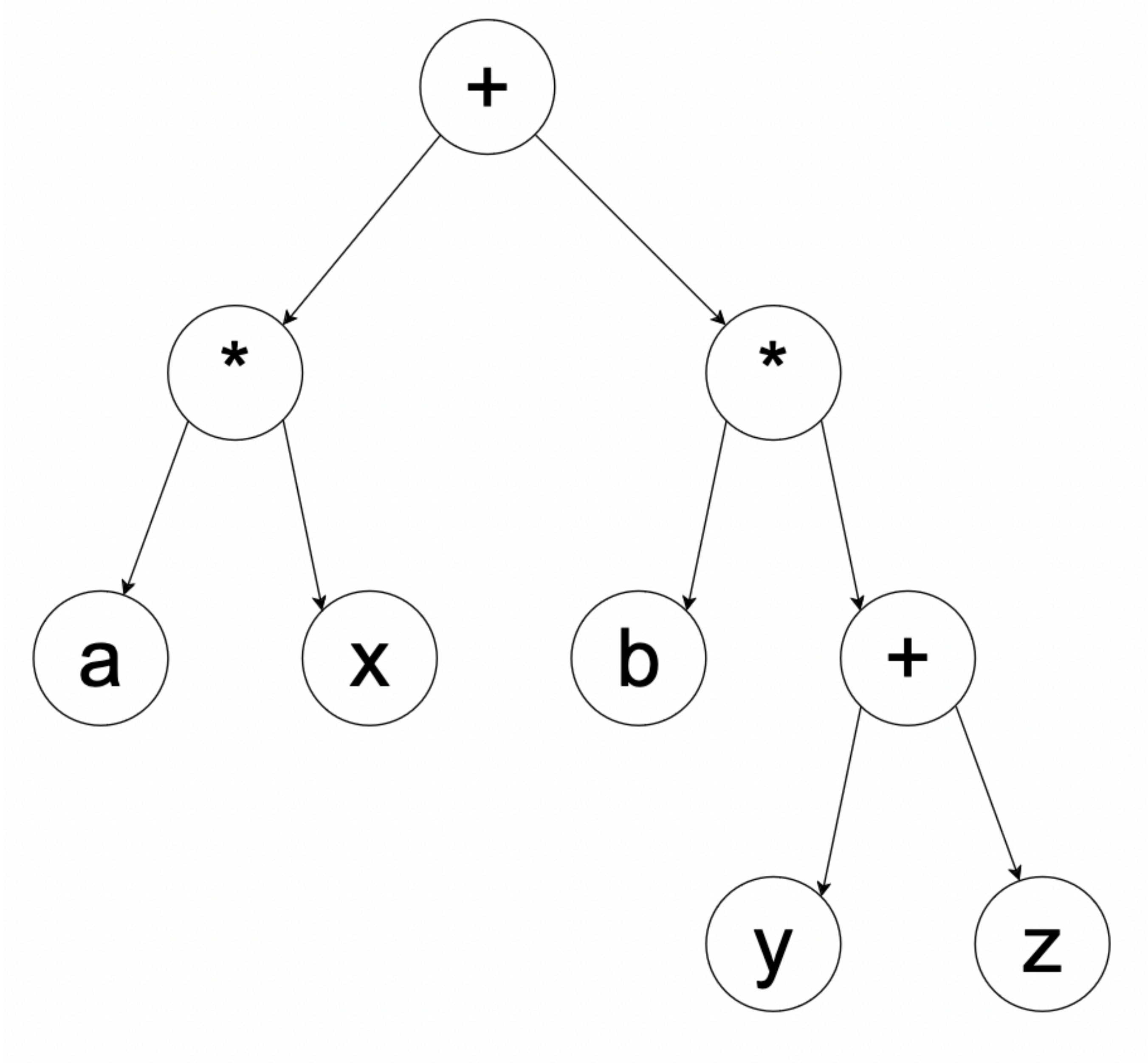
Solvers presolve



Going beyond state of the art

Idea 2: Using syntactic formulation for presolve

Modeling tools

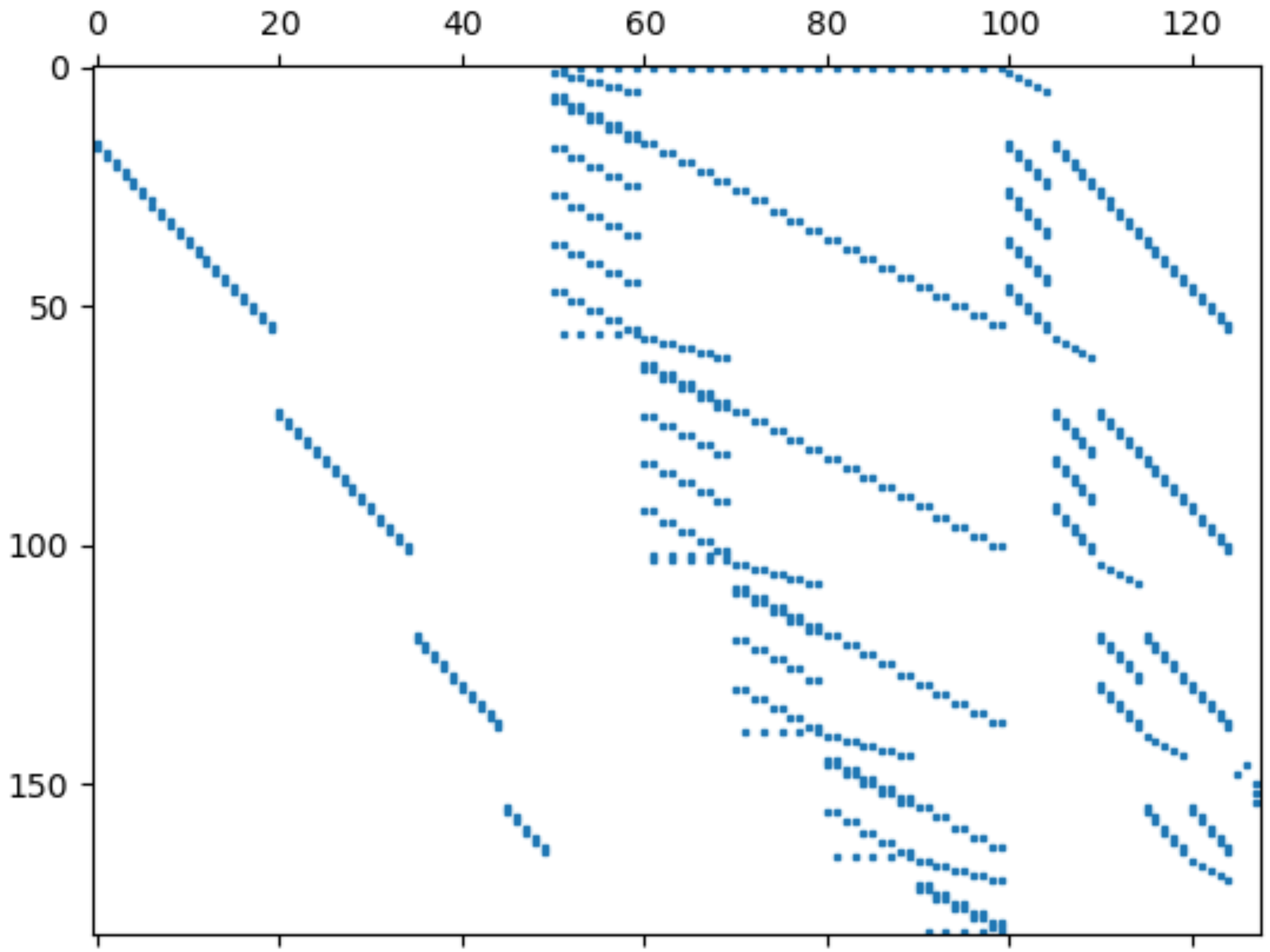


- Work directly on the symbolic syntax trees to simplify the model
  - Improving the solving time

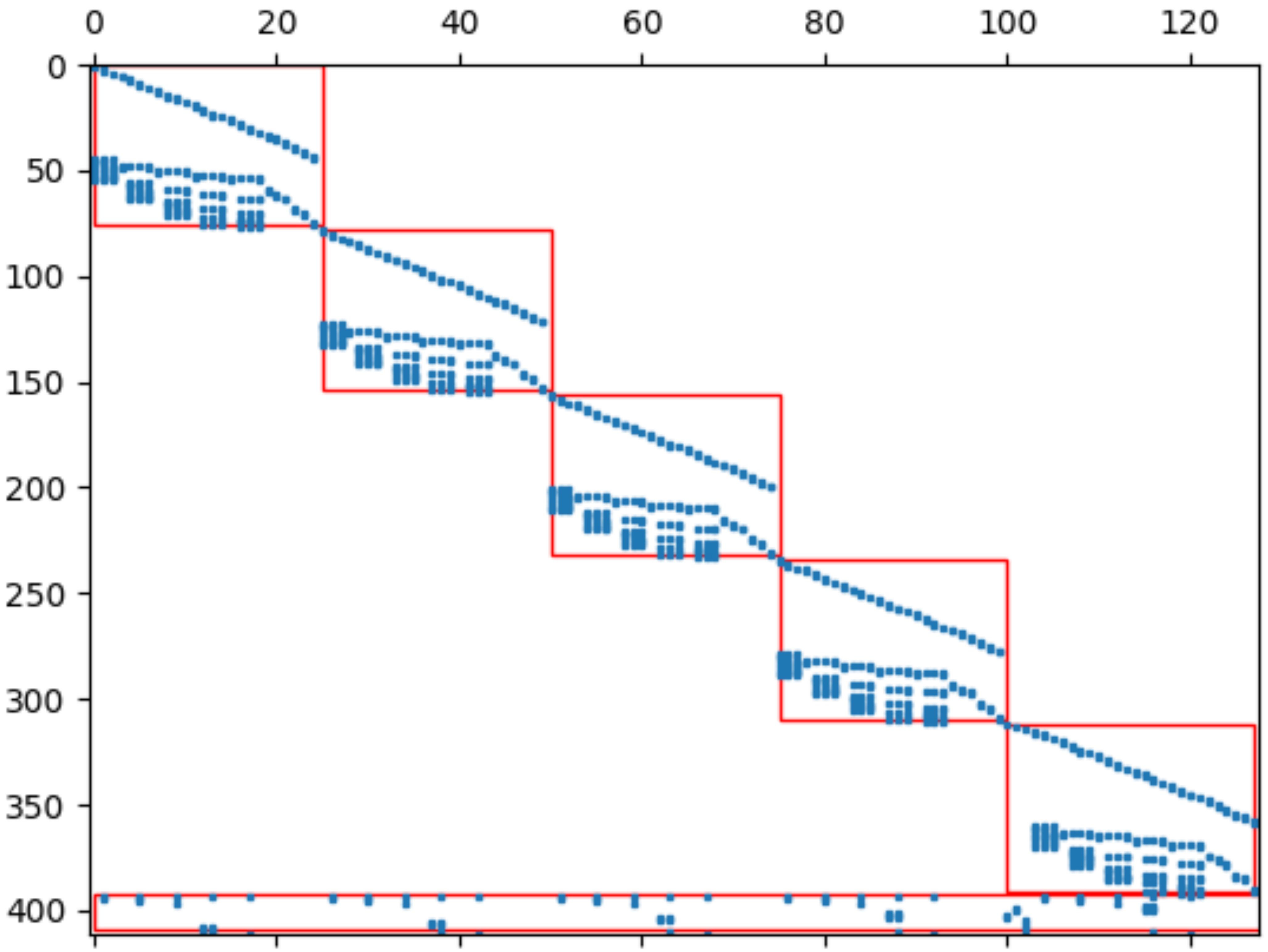
Going beyond state of the art

Idea 3: Using structure for solving

Solving time:



Solved in 25 seconds by Gurobi



Solved in 2.5 seconds using Dantzig-Wolfe

Going beyond state of the art

Idea 4: LLMs

- Input: description of the model
- Output: the MILP reformulation in a modeling language
- Needed expertise:
  - To deal with hallucinations
  - Stronger reformulations
  - …

Creating a modeling tool

Summary

- Step 1: Know your community
- Step 2: List your requirements
- Step 3: Find a good fit

Creating a modeling tool

Summary

- Step 1: Know your community
- Step 2: List your requirements
- Step 3: Find a good fit
- You should stop here

Creating a modeling tool

Summary

- Step 1: Know your community
- Step 2: List your requirements
- Step 3: Find a good fit
- Step 4: Find your niche
- Step 5: Make the tool
- Exciting times for modeling tools and a lot of further work