Exploiting structure in MILP: a modeler's perspective

SIAM Conference on Optimization (OP23)

Seattle, USA

May 31- June 3, 2023

Bardhyl Miftari, Mathias Berger, Guillaume Derval and Damien Ernst

Table of content

- Structured MILPs
- MILP workflow
- Modeling tools
- GBOML
- Benchmarking
- Conclusion

Structure in MILPs

Examples

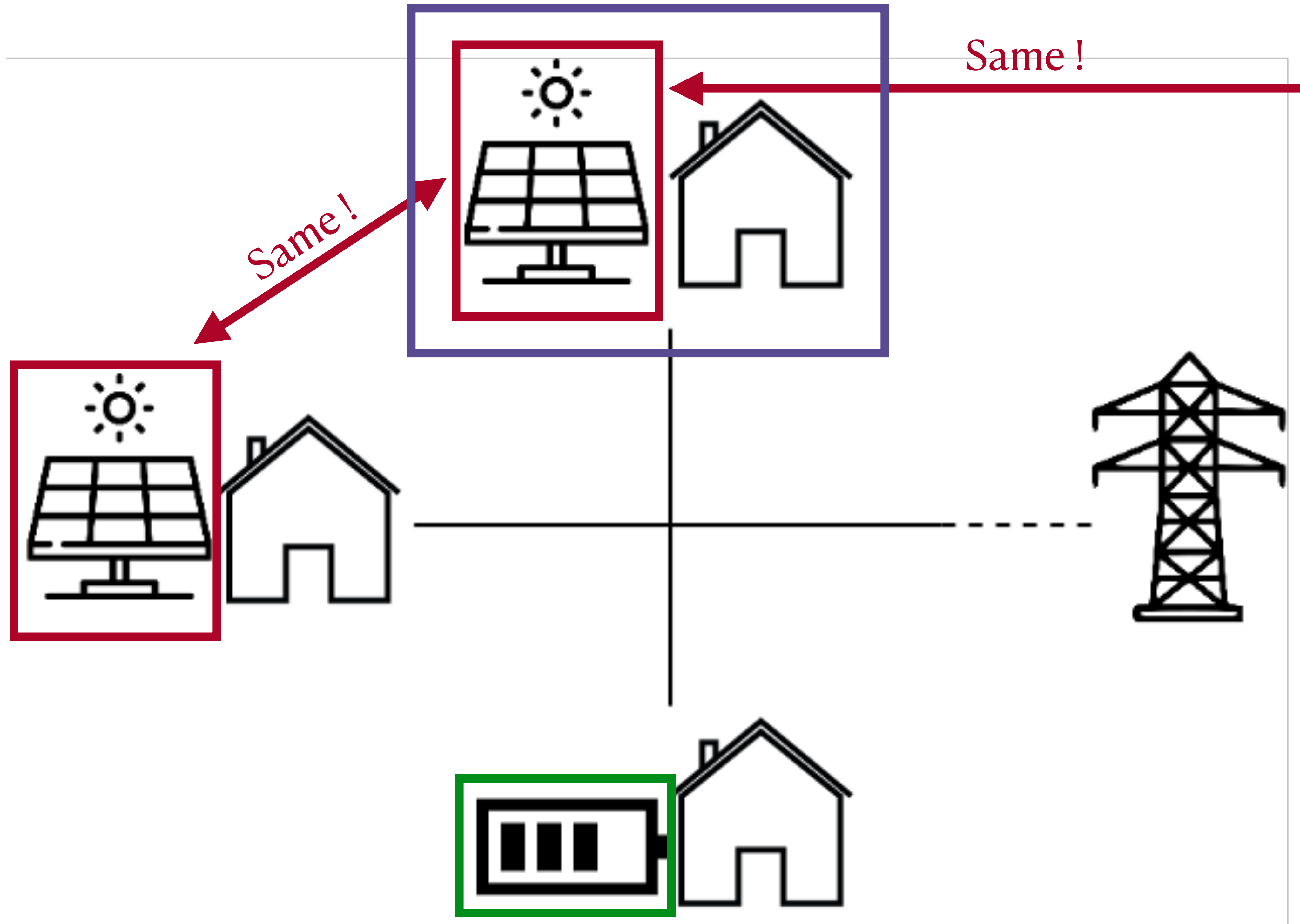


Figure 1: renewable energy community

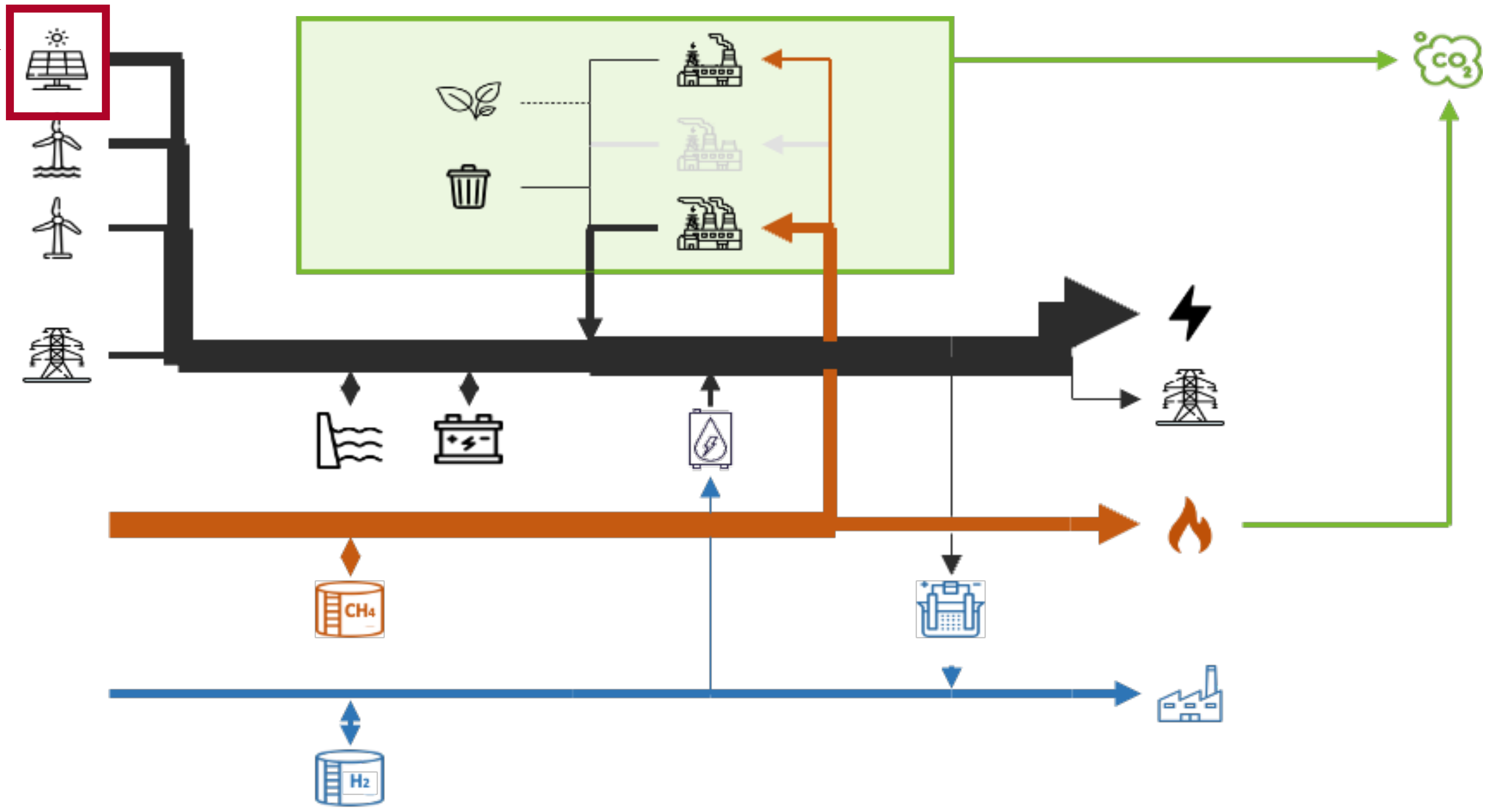


Figure 2: Belgian energy model

- Network of components
- Similar components re-used several times
- Frequent in energy or supply chain optimization

How to exploit this structure ?

MILP Workflow

Basics

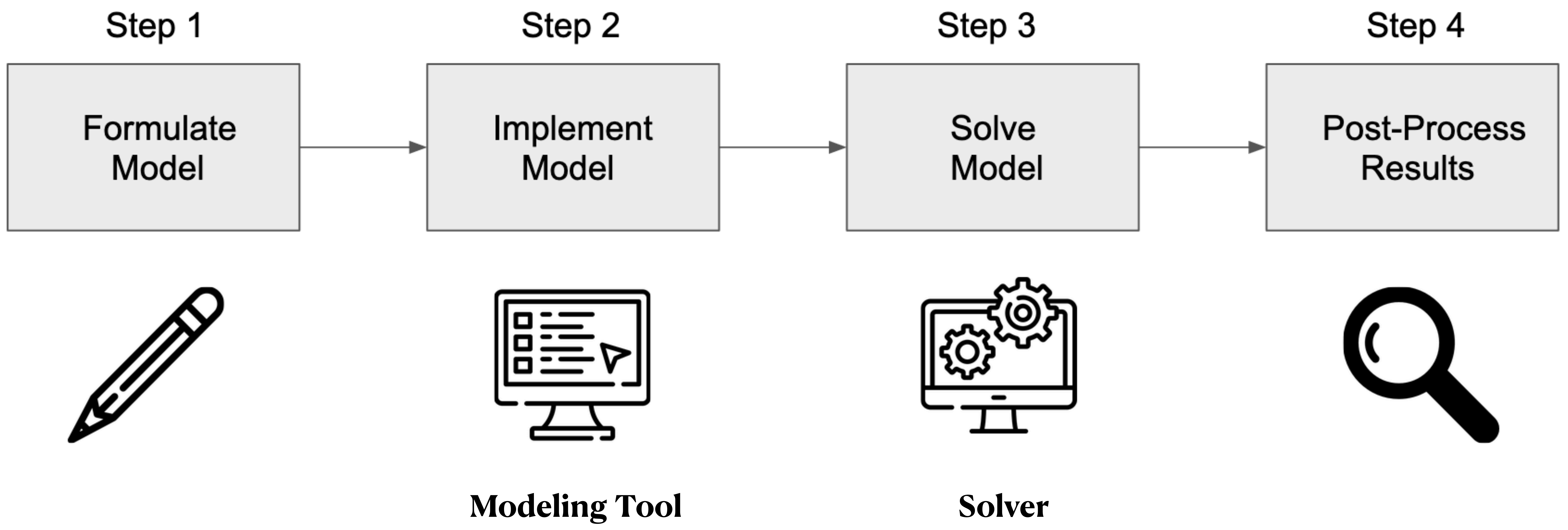


Figure 3: MILP workflow

MILP Workflow

Basics

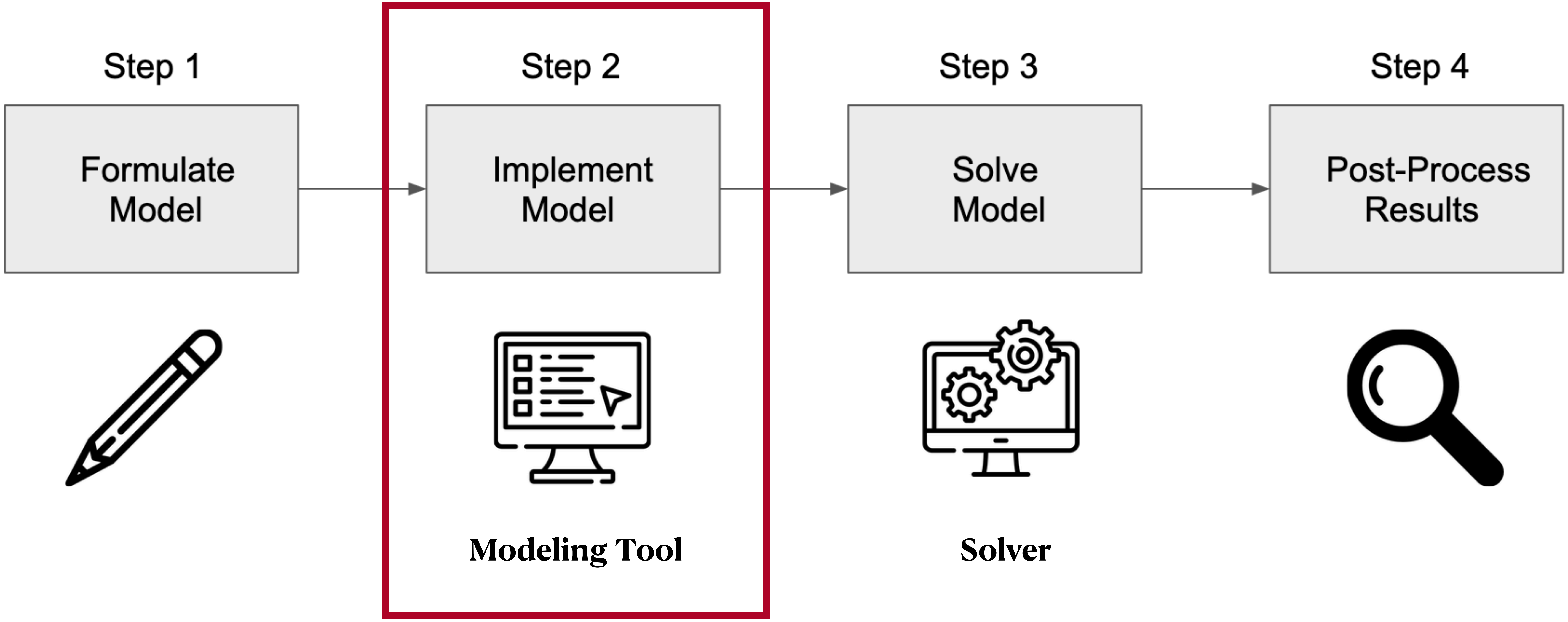


Figure 3: MILP workflow

Modeling Tools

Basics

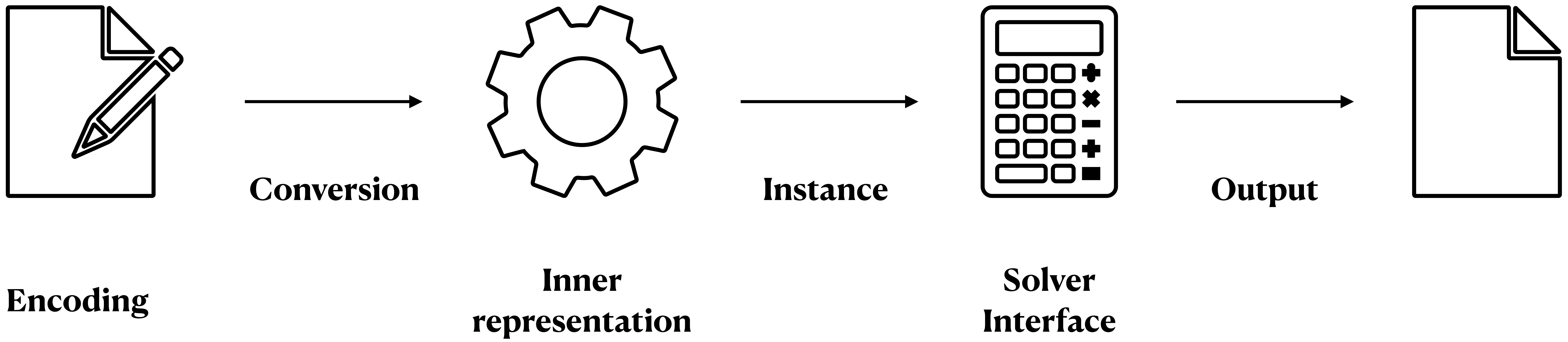


Figure 4: Modelling tools workflow

Modeling Tools

Basics

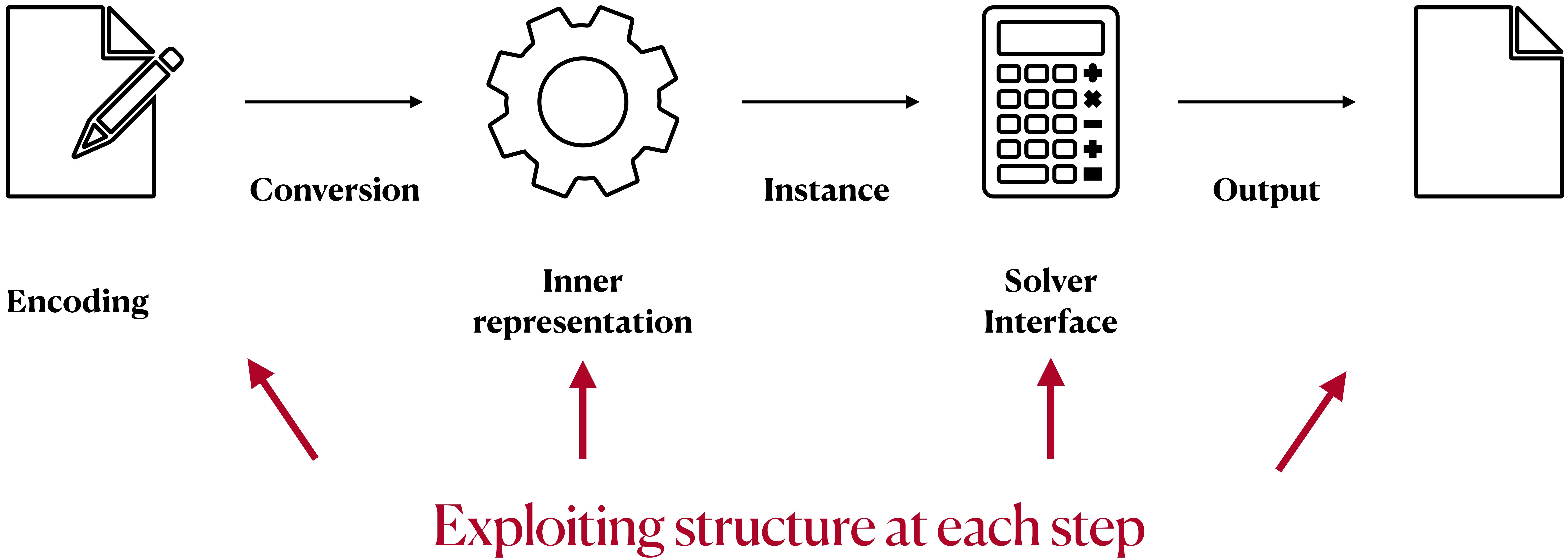


Figure 4: Modelling tools workflow

Modeling Tools

AMLs

- **A**lgebraic **M**odeling **L**anguages (**AMLs**)
  - Formulation close to mathematical notation
  - Very expressive (e.g. can represent any mixed-integer nonlinear program)
  - Often interface with multiple solvers
  - Do not exploit structure encoding in their basic form
  - Examples:

GAMS [1] JuMP [2] AMPL STREAMLINED MODELING FOR REAL OPTIMIZATION [3] pulp [4] PYOMO [5]

Modeling Tools

OOMEs

- **O**bject-**O**riented **M**odeling **E**nvironments (**OOMEs**)
  - Focus on one particular application (e.g. energy system sizing and operations)
  - Usually make use of predefined components that are “imported”
  - Difficult to add or modify the components
  - Typically have advanced data processing capabilities tailored to the application
  - Examples:

PyPSA [6]

Calliope [7]

PLEXOS [8]

BALMOREL Energy system model [9]

solph ['sɒlv] open energy modelling framework [10]

Dispa-SET Power system modelling [11]

Can we go further ?

Going Further

GBOML

- The **G**raph-**B**ased **O**ptimization **M**odeling **L**anguage (**GBOML**)[12-13] combines the strengths of AMLs and OOMEs
  - Open-Source and Stand-alone
  - Can represent any MILP
  - Exploits structure in various ways
  - Syntax close to the mathematical notation
  - Time-indexed models can be encoded easily
  - Allows component definition, re-use and component assembling
  - Interfaces with various solvers

Going Further

GBOML

- In GBOML, structure is exploited at all levels:

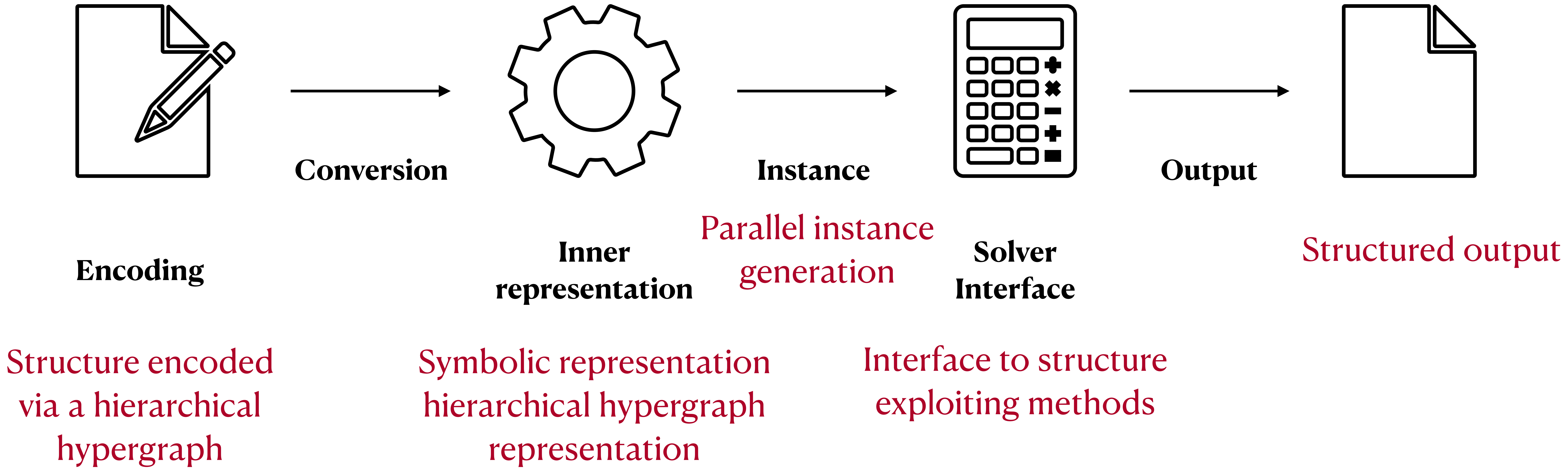


Figure 5: GBOML structure exploiting workflow

GBOML abstraction

Hierarchical hypergraph

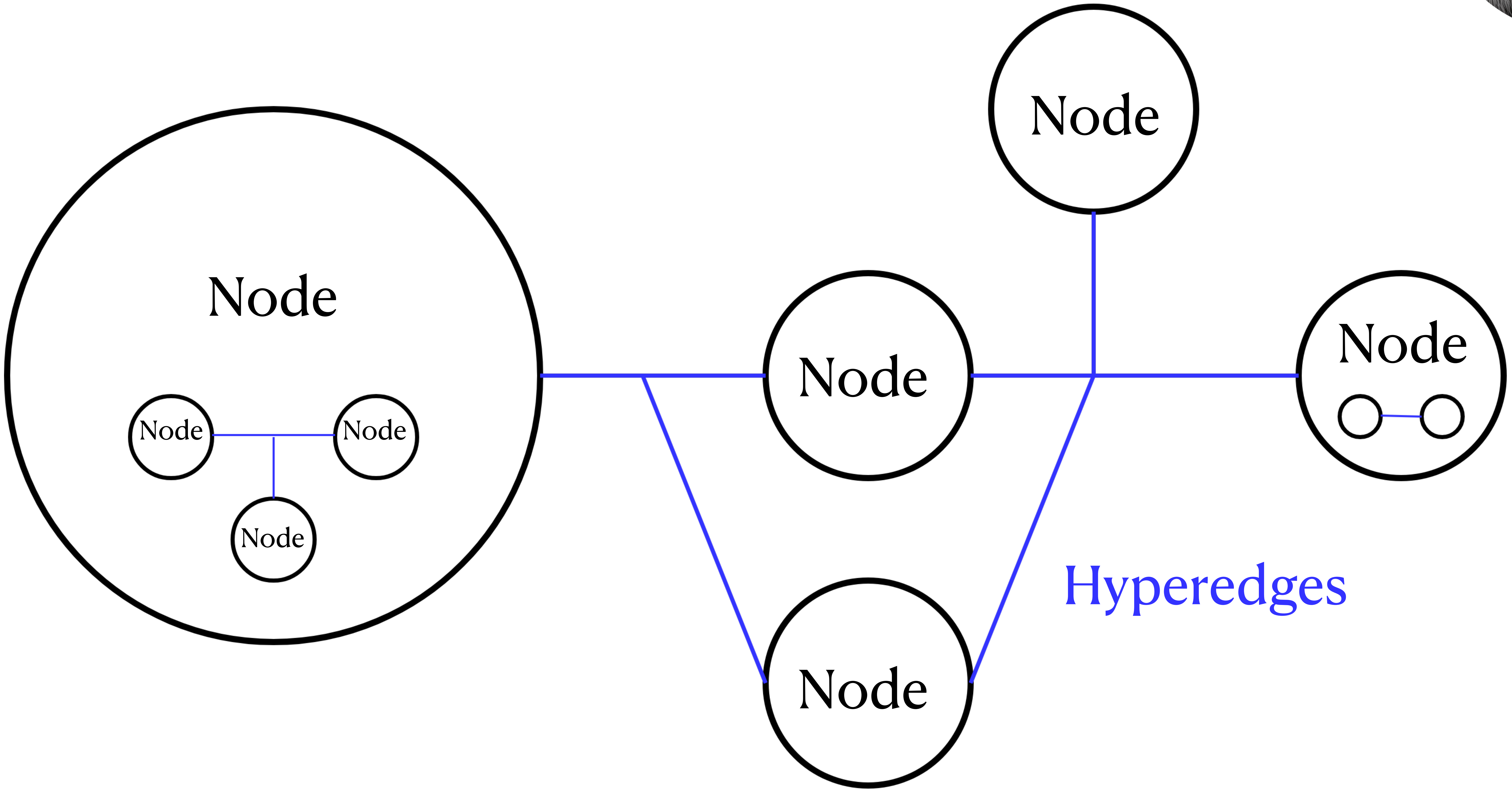


FIGURE 6 : Representation of one particular hierarchical hypergraph made-up of 5 nodes and 2 hyperedges. The node most to the left and to the right both contain a hypergraph themselves.

GBOML language

Basics

```
#TIMEHORIZON
T = <value>;
```

```
#NODE <node_name>
 #PARAMETERS
  <param_def>
 #VARIABLES
  <var_def>
 #CONSTRAINTS
  <constr_def>
 #OBJECTIVES
  <obj_def>
```

```
#HYPEREDGE <edge_name>
 #PARAMETERS
  <param_def>
 #CONSTRAINTS
  <constr_def>
```

GBOML language

Basics

```
#TIMEHORIZON
T = <value>;
```

```
#NODE <node_name>
 #PARAMETERS
  <param_def>
 #VARIABLES
  <var_def>
 #CONSTRAINTS
  <constr_def>
 #OBJECTIVES
  <obj_def>
```

```
#HYPEREDGE <edge_name>
 #PARAMETERS
  <param_def>
 #CONSTRAINTS
  <constr_def>
```

GBOML language

Basics

```
#TIMEHORIZON
T = <value>;
```

```
#NODE <node_name>
 #PARAMETERS
  <param_def>
 #VARIABLES
  <var_def>
 #CONSTRAINTS
  <constr_def>
 #OBJECTIVES
  <obj_def>
```

```
#HYPEREDGE <edge_name>
 #PARAMETERS
  <param_def>
 #CONSTRAINTS
  <constr_def>
```

GBOML inner structure

Hierarchical hypergraph

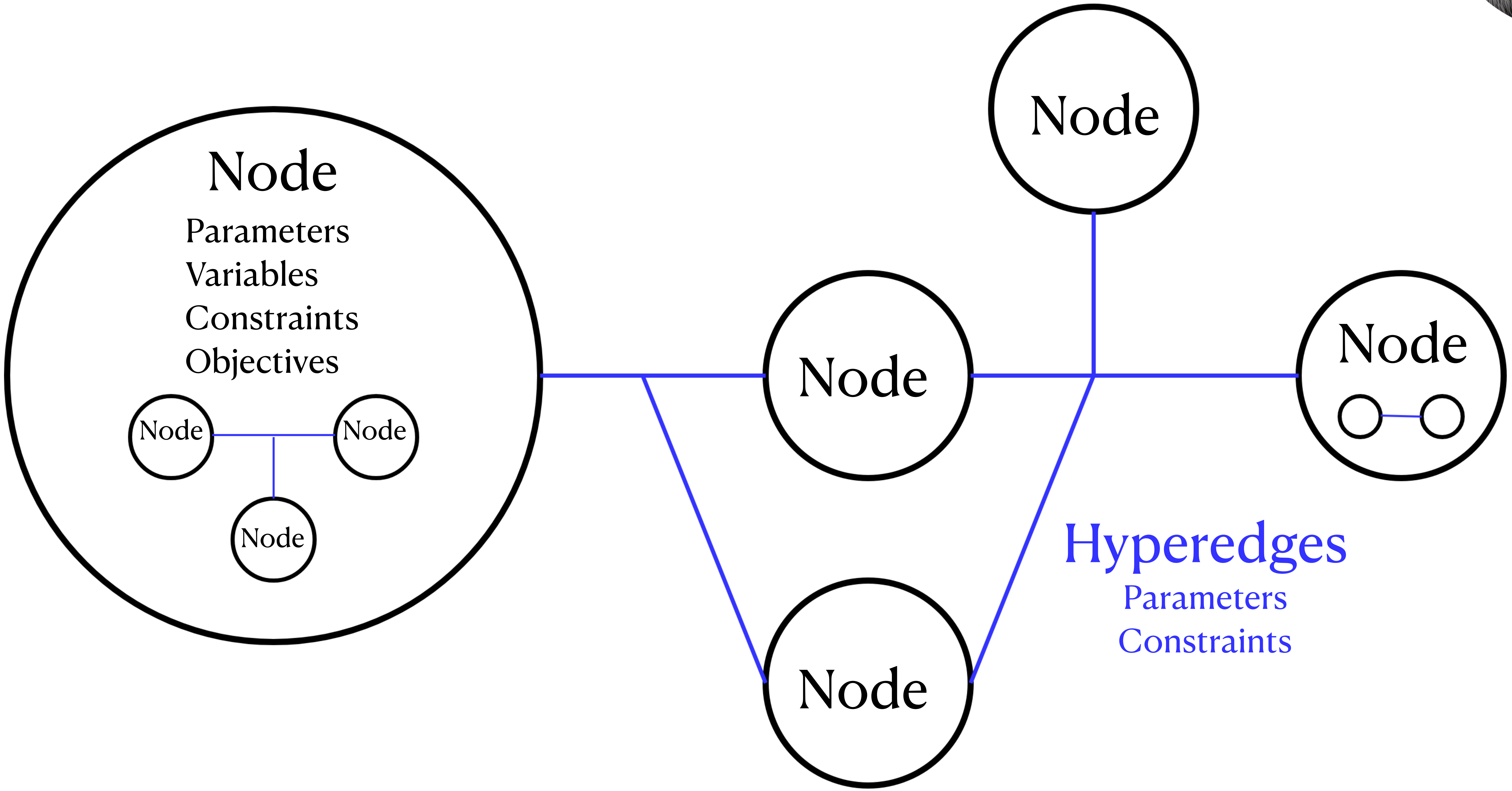
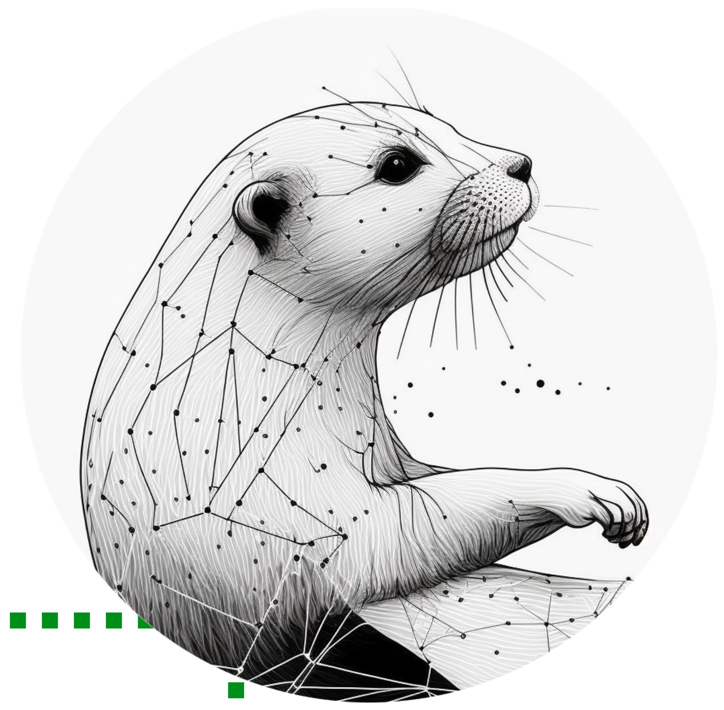
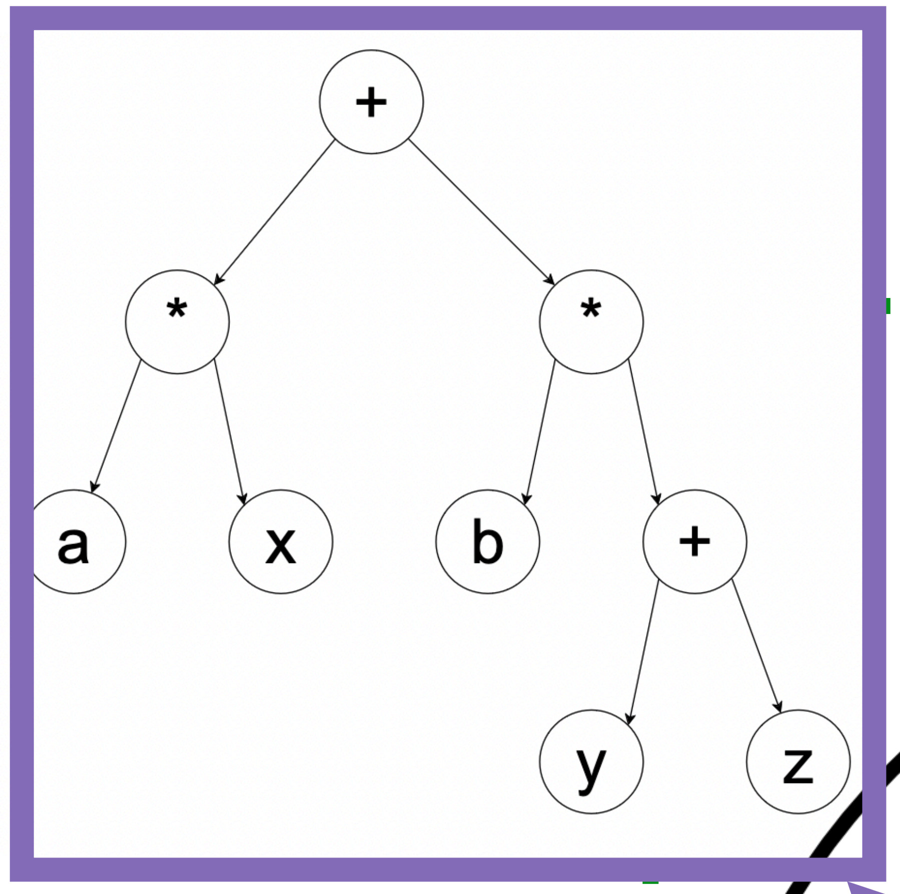


FIGURE 6 : Representation of one particular hierarchical hypergraph made-up of 5 nodes and 2 hyperedges. The node most to the left and to the right both contain a hypergraph themselves.

GBOML inner structure



Hierarchical hypergraph



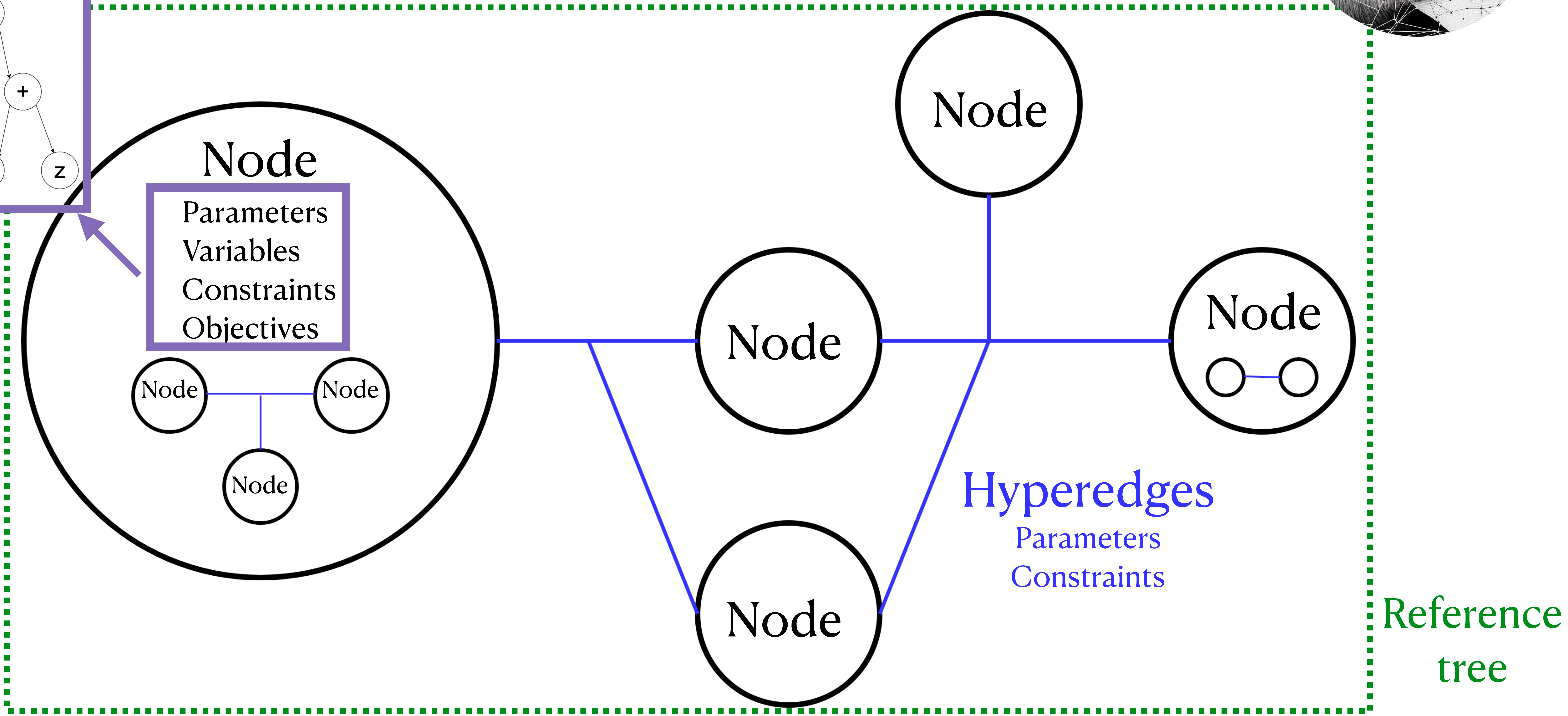


FIGURE 6 : Representation of one particular hierarchical hypergraph made-up of 5 nodes and 2 hyperedges. The node most to the left and to the right both contain a hypergraph themselves.

GBOML inner structure

Hierarchical hypergraph

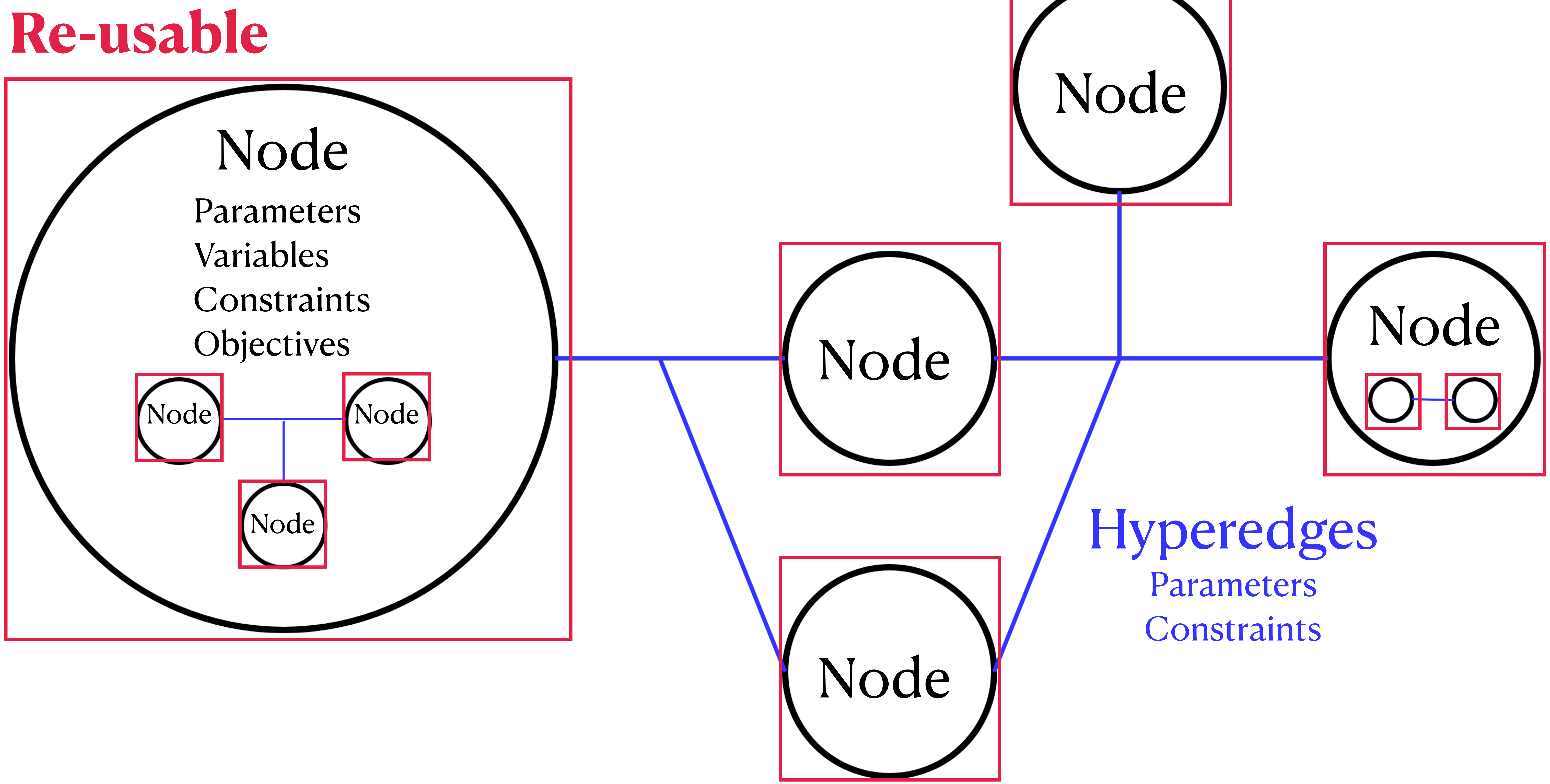


FIGURE 6 : Representation of one particular hierarchical hypergraph made-up of 5 nodes and 2 hyperedges. The node most to the left and to the right both contain a hypergraph themselves.

GBOML inner structure

Hierarchical hypergraph

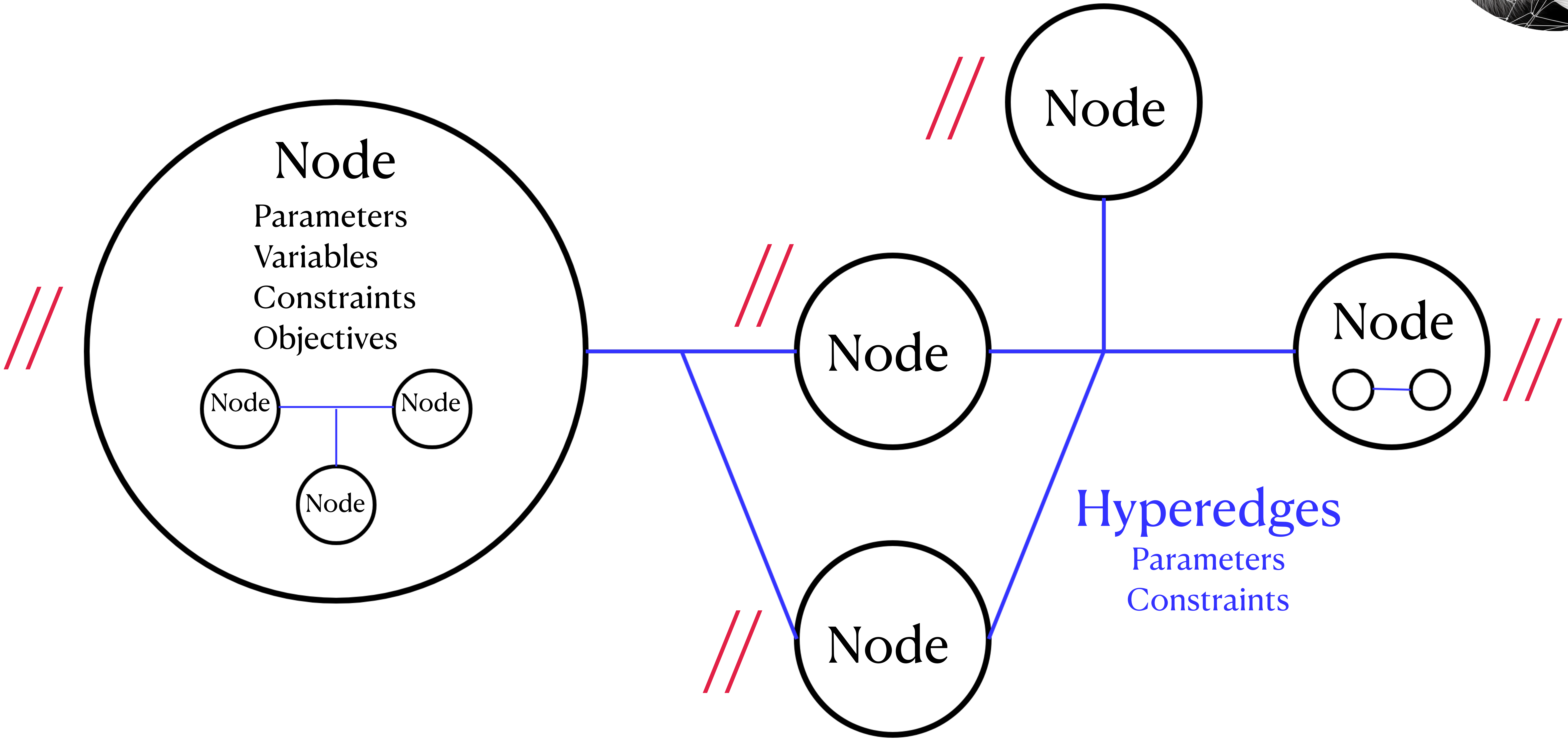


FIGURE 6 : Representation of one particular hierarchical hypergraph made-up of 5 nodes and 2 hyperedges. The node most to the left and to the right both contain a hypergraph themselves.

GBOML solver interface

Methods

- Commercial solvers

GUROBI OPTIMIZATION [14] FICO [15] IBM [16]

- Open-source solvers

HiGHS [17] COIN|OR [18]

- Structure exploiting methods
    - DSP[19]: Dantzig-Wolfe decomposition
    - CPLEX: Benders decomposition

Benchmarking

Benchmarking

Instance generation time

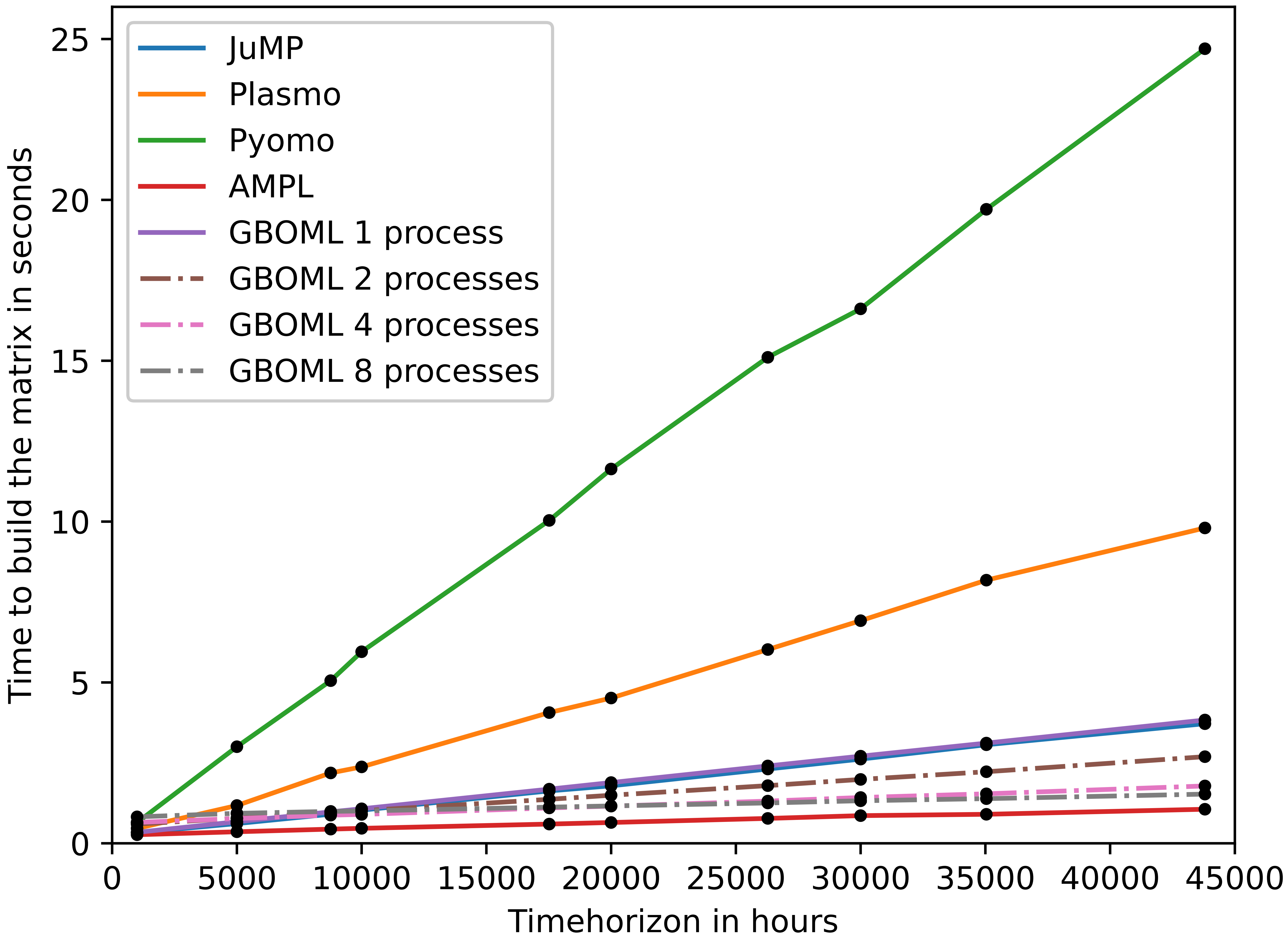


FIGURE 7 : time to generate a remote renewable energy hub instance[20]
for a growing time horizon with different tools

Benchmarking

Peak RAM usage

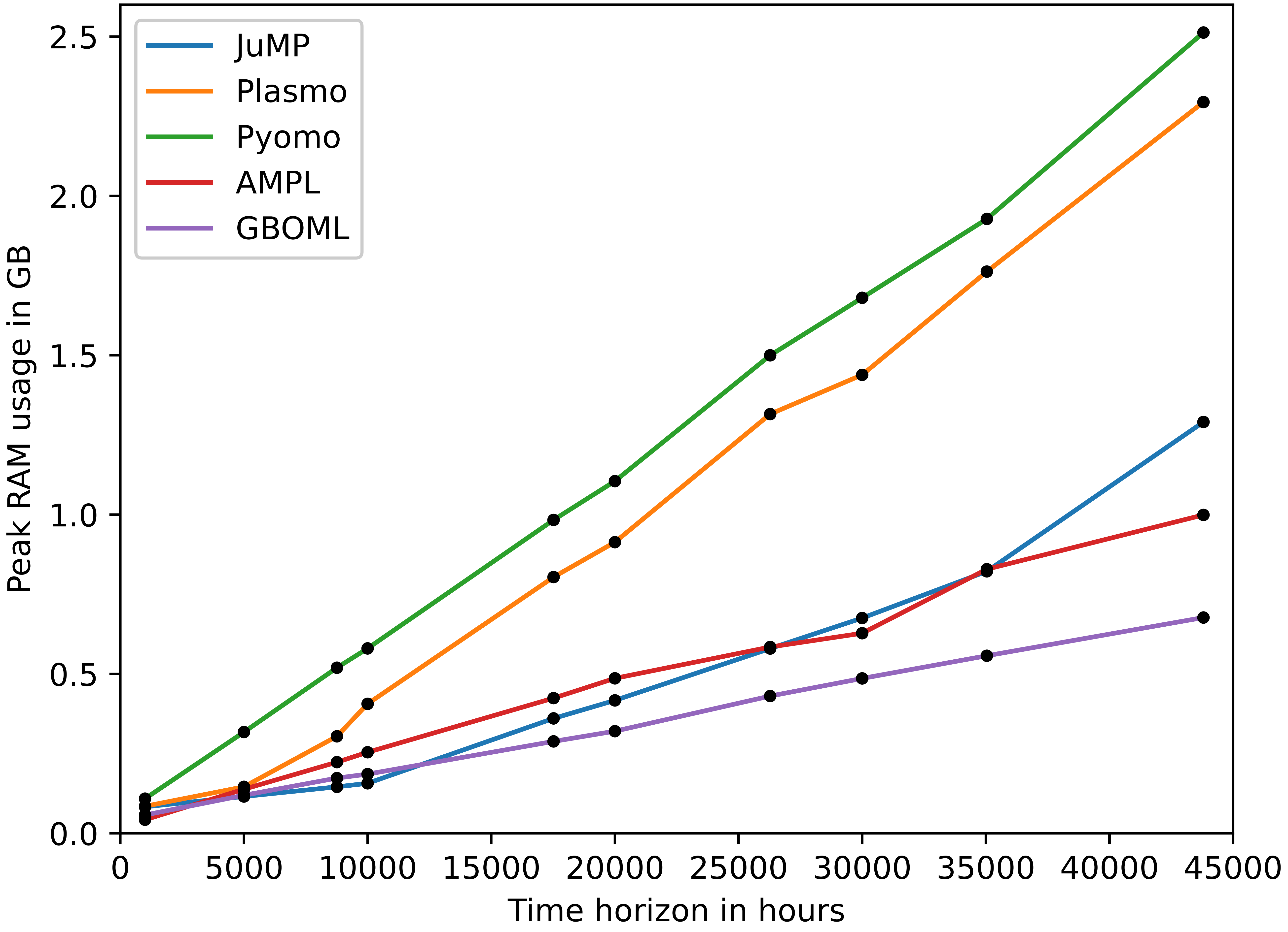
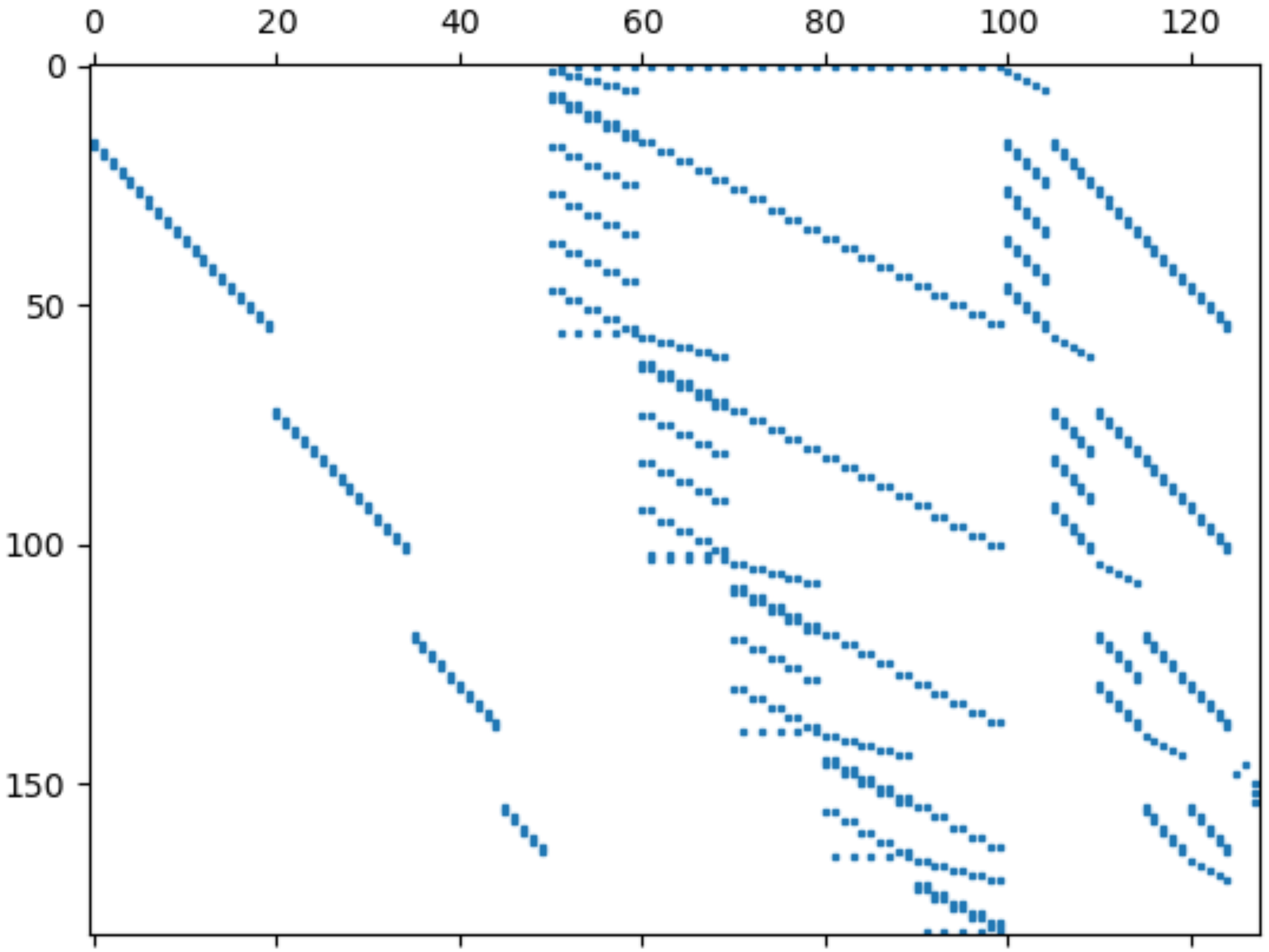


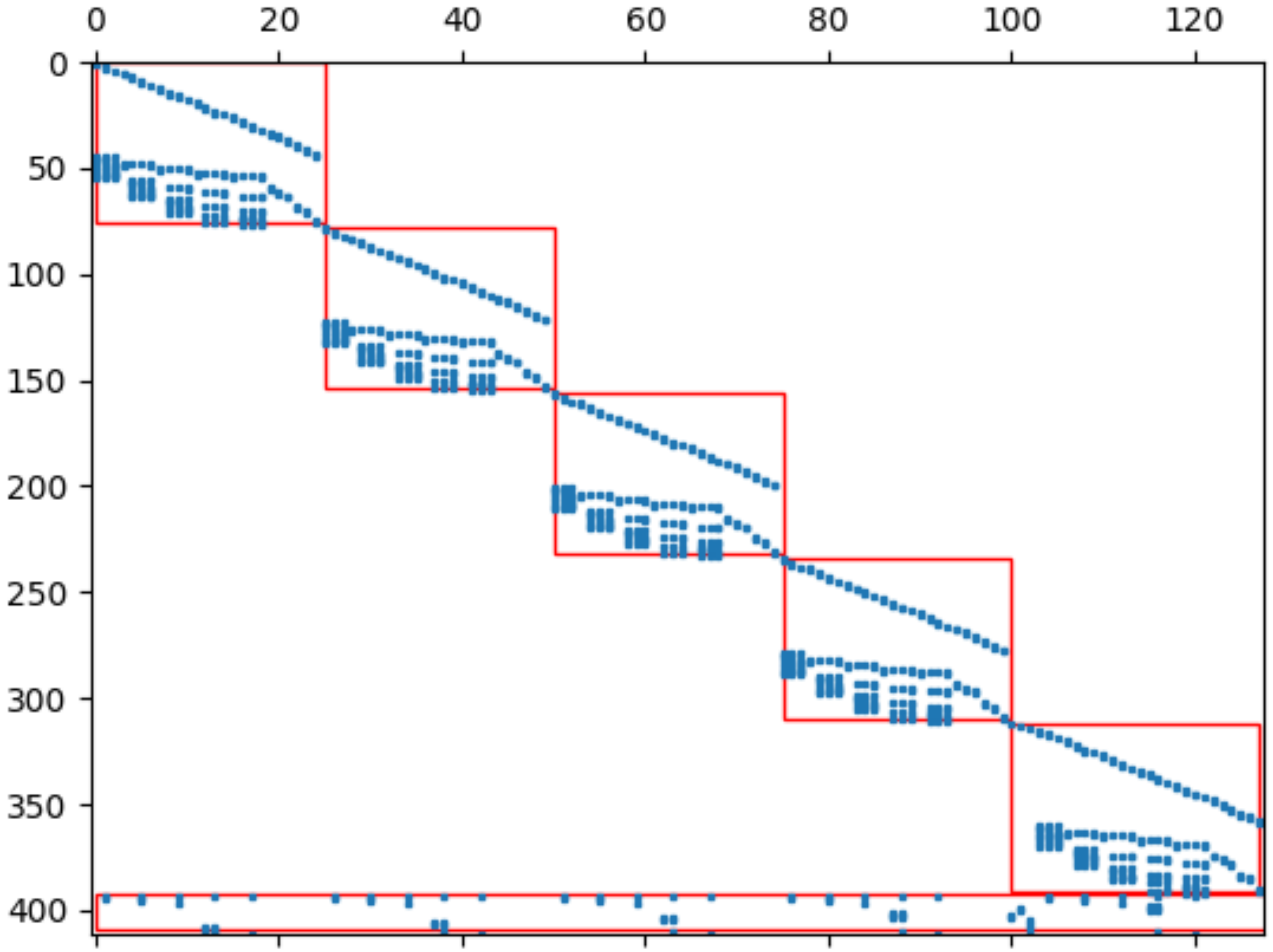
FIGURE 8 : peak ram usage of generating a remote renewable energy hub instance[20]
for a growing time horizon with different tools

Benchmarking

Solving time



Solved by Gurobi in 25 seconds



Solved using Dantzig-Wolfe in 2.5 seconds

FIGURE 9 : No swot problem from the MIPLIB[21] representation without structure in a MPS file (left) and with structure in GBOML (right)

Conclusion

Exploiting structure in MILPs

- Structure can be used to
  - encode problems in a more «natural» way
  - allow component definition, re-use and model assembling
  - generate instances of problems faster and use less RAM
  - parallelize model generation
  - interface with structure exploiting methods
    - which can sometimes lead to faster solving time

Acknowledgments

- We would like to thank
  - SPF Economie (Federal government of Belgium)[30] for their financial support through the INTEGRATION project
  - The Walloon Region for their financial support through the INTEGCER project on renewable energy communities
  - SIAM OP23 for the opportunity of presenting our work and the organization

References

[1] The General Algebraic Modeling Language, GAMS. https://www.gams.com/

[2] JuMP, https://jump.dev/JuMP.jl/stable/

[3] A Mathematical Programming Language, AMPL. https://ampl.com/

[4] Pulp. https://github.com/coin-or/pulp

[5] Pyomo. http://www.pyomo.org/

[6] PyPSA, Python for Power System Analysis. https://pypsa.org/

[7] Calliope. Calliope: a multi-scale energy systems modelling framework. https://calliope.readthedocs.io/en/stable/#

[8] Plexos, The Energy Analytics and Decision Platform for all Systems. https://www.energyexemplar.com/plexos

[9] Balmorel, http://www.balmorel.com/

[10] oemof.solph, https://github.com/oemof/oemof-solph

[11] The Dispa-SET model. http://www.dispaset.eu/en/latest/

[12] Bardhyl Miftari et al., "GBOML: Graph-Based Optimization Modeling Language", https://joss.theoj.org/papers/10.21105/joss.04158, 2022

[13] Bardhyl Miftari et al., "GBOML: a Structure-exploiting Optimization Modeling Language in Python", https://orbi.uliege.be/handle/2268/296930, 2022

[14] Gurobi Optimization, LLC. All Rights Reserved. https://www.gurobi.com/

[15] FICO® Xpress Optimization. https://www.fico.com/en/products/fico-xpress-optimization

[16] IBM ILOG CPLEX Optimizer. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer

[17] HiGHS - high performance software for linear optimization. https://highs.dev/

[18] CBC/CLP from COIN-OR Foundation, Inc..https://www.coin-or.org/

[19] DSP, Argonne National Laboratory. https://github.com/Argonne-National-Laboratory/DSP

[20] Mathias Berger et al., "Remote Renewable Hubs for Carbon-Neutral Synthetic Fuel Production", in Frontiers in Energy Research 9 (2021), p.200. DOI 10.3389/fenrg.2021.671279. https://www.frontiersin.org/article/10.3389/fenrg.2021.671279

[21] MIPLIB, https://miplib.zib.de/