GBOML: A Modelling Tool for Structured MILPs

Bardhyl Miftari[1,*], **Mathias Berger**[1], **Guillaume Derval**[1], **Quentin Louveaux**[1], **Damien Ernst**[1,2]

*bmiftari@uliege.be, [1]University of Liège, Belgium, [2]LTCI, Telecom Paris, Institut Polytechnique de Paris, France

Yet another modelling tool ?

GBOML aims to bridge the gap between AMLs and OOMEs.

Algebraic Modelling Languages (AMLs) & Object-Oriented Modelling Environments (OOMEs) ⇒ The Graph-Based Optimization Modelling Language (GBOML)

Focus on mathematical modelling: Close to math notation; Very expressive; No structure exploitation

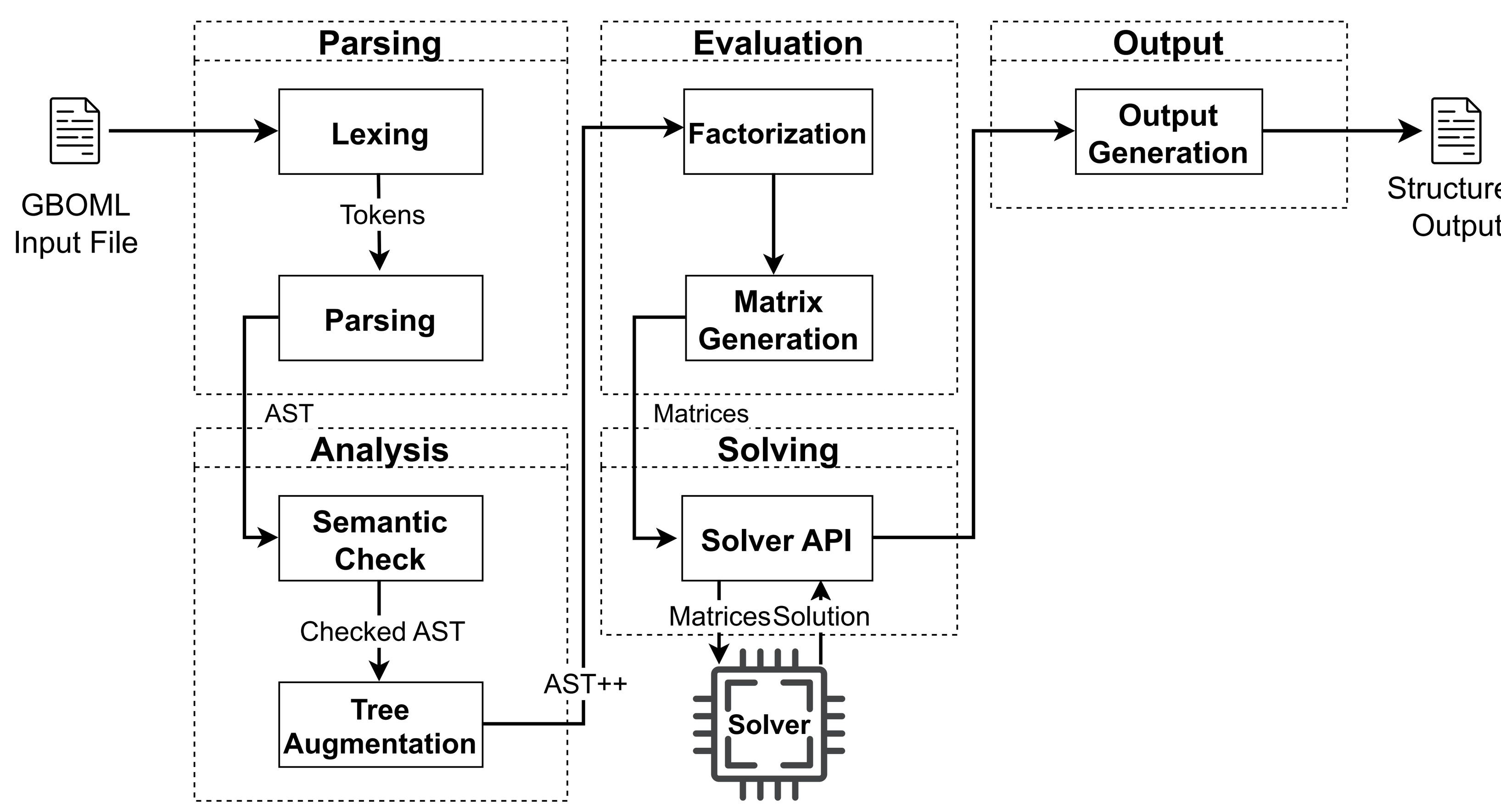
Problem-specific modelling tools: Reusable components; Component assembling; Enable structure encoding; Difficult to extend

Mix of AMLs and OOMEs: Close to math notation; Very expressive; Component assembling; Reusable, easy to create components; Exploits structure

Structured MILPs

- Arise in many applications such as **energy system planning** and **supply chain management** problems
- Often possess a **time-index**
- Can often be seen as **networks of components or units**
- Can often be encoded by a **hierarchical hypergraph**

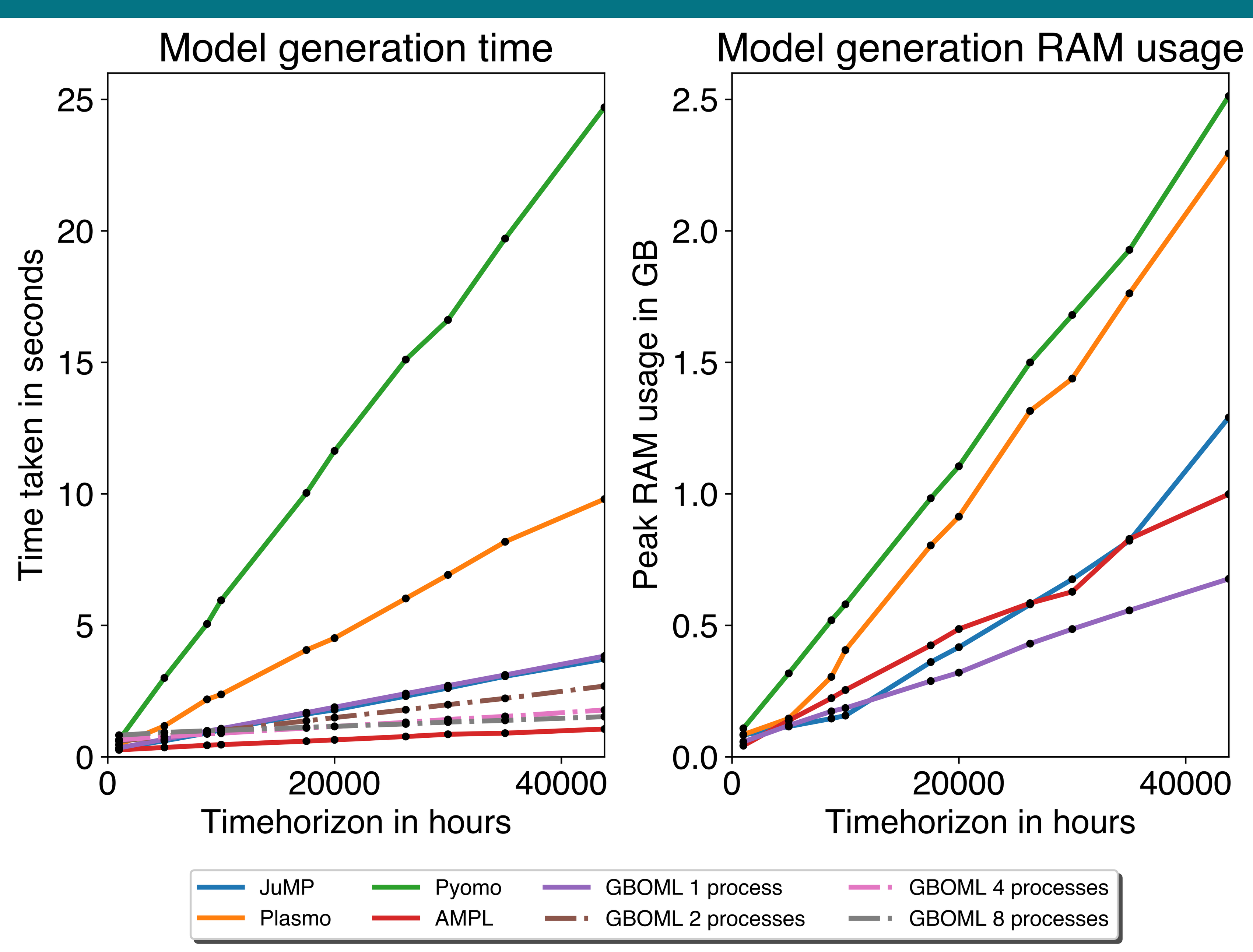
GBOML



The **Graph-Based Optimization Modelling Language (GBOML)[1, 2]**

- is **open-source** and coded in **Python** (available on PyPI)
- relies on a **hierarchical hypergraph abstraction** to capture structure
- interfaces with both commercial and open-source **solvers**
- **exploits structure** in
  - **model encoding** via its hypergraph abstraction
  - **model generation** via its inner representation, vectorization and parallel model generation
  - **model solving** by interfacing with structure exploiting methods (Dantzig-Wolfe and Benders decomposition)

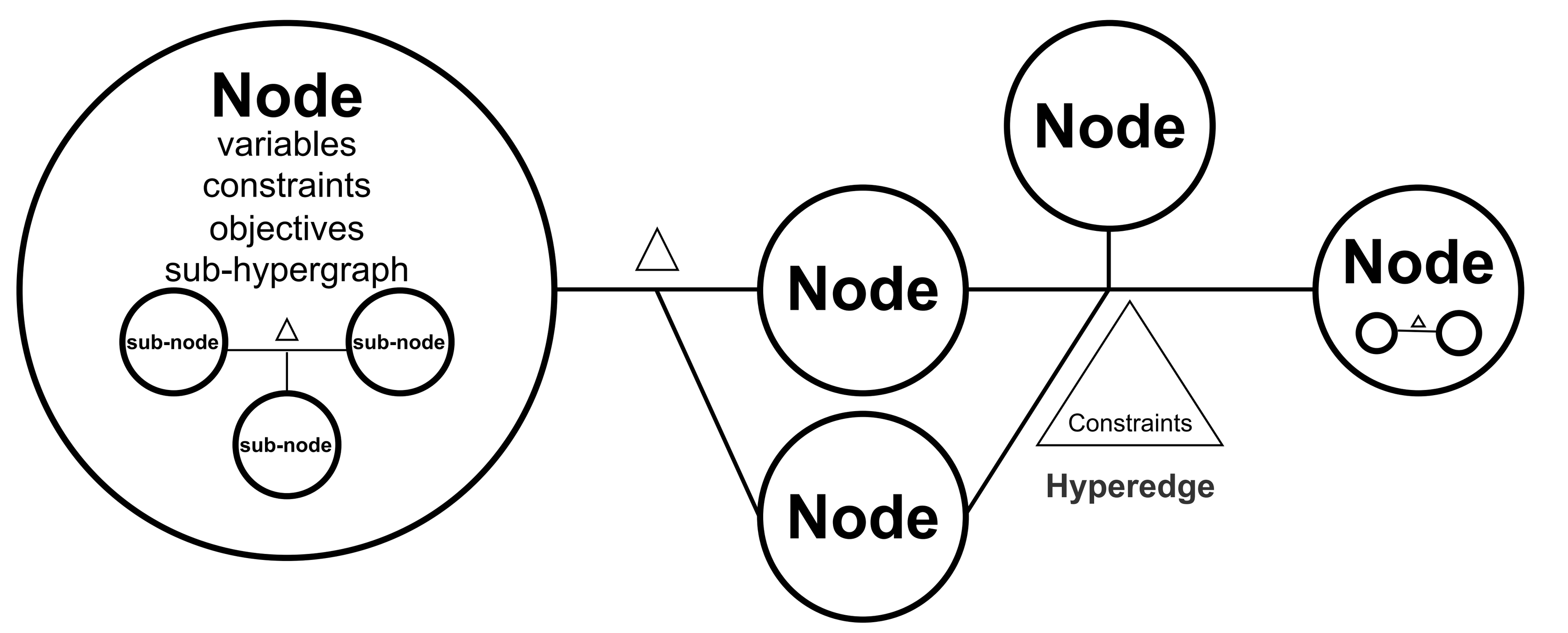
Benchmark



Mathematical formulation

This work focuses on block-decomposable problems that can be encoded by a **hierarchical hypergraph** $G = \langle \bullet, \blacktriangle \rangle$, where

- $\bullet$ is the set of nodes
- $\blacktriangle$ is the set of hyperedges linking these nodes.



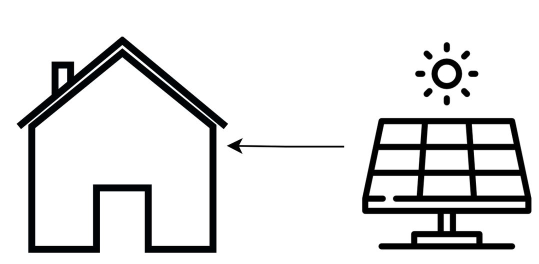
Each node $\bigcirc \in \bullet$ is made up of variables, objectives $obj_{\bigcirc}$, constraints $cstr_{\bigcirc}$ that need to be satisfied and a sub-hypergraph $G_{\bigcirc} = \langle \bullet_{\bigcirc}, \blacktriangle_{\bigcirc} \rangle$.
Each hyperedge $\triangle \in \blacktriangle$ is made up of constraints $cstr_{\triangle}$ that connect nodes' variables. The overall problem $P(G)$ is written as,

$$P(G) \equiv \min \sum_{\bigcirc \in \bullet} f(\bigcirc) \quad \text{s.t.} \quad g(\bigcirc) \text{ is true } \forall \bigcirc \in \bullet, \quad cstr_{\triangle} \text{ is true } \forall \triangle \in \blacktriangle$$

$$f(\bigcirc) = obj_{\bigcirc} + \sum_{o \in \bullet_{\bigcirc}} f(o),$$

$$g(\bigcirc) = cstr_{\bigcirc} \wedge \left[g(o)\ \forall o \in \bullet_{\bigcirc} \right] \wedge \left[cstr_{\triangle}\ \forall \triangle \in \blacktriangle_{\bigcirc} \right]$$

Example



We consider a house that wants to minimize its overall electricity bill by installing PV panels. First, we model the PV panels in GBOML by writing,

```
#NODE PV
  #PARAMETERS
      cost_invest = 120;
      cost_op = 1;
      irradiance = import "irradiance.csv";
      max_capacity = 500.0;
  #VARIABLES
      internal: capacity;
      external: electricity[T];
  #CONSTRAINTS
      electricity[t] <= irradiance[t] * capacity;
      capacity <= max_capacity;
      capacity >= 0;
      electricity[t] >= 0;
  #OBJECTIVES
      min: cost_invest * capacity;
      min: cost_op * electricity[t];
```

We can then import the node PV and write the overall problem as,

```
#TIMEHORIZON T = 24*365*5;

#NODE HOUSE
  #PARAMETERS
    demand = import "demand.csv";
    energy_price = 2;
  #NODE PV = import "PV" from "PV.gboml";
  #VARIABLES
    external: tobuy[T];
    internal: panels[T] <- PV.electricity[T];
  #CONSTRAINTS
    tobuy[t] >= demand[t] - panels[t];
    tobuy[t] >= 0;
  #OBJECTIVES
    min: tobuy[t];
```

[1] Bardhyl Miftari et al. "GBOML: A Structure-Exploiting Optimization Modelling Language in Python". 2022. URL: https://gitlab.uliege.be/smart_grids/public/gboml.

[2] Bardhyl Miftari et al. "GBOML: Graph-Based Optimization Modeling Language". In: *Journal of Open Source Software* 7.72 (2022), p. 4158. DOI: 10.21105/joss.04158. URL: https://doi.org/10.21105/joss.04158.