GBOML: A MODELLING TOOL FOR STRUCTURED MILPs
Bardhyl Miftari1,*, Mathias Berger1, Guillaume Derval1, Quentin Louveaux1, Damien Ernst1,2
1University of Liège, Belgium, 2LTCI, Telecom Paris, Institut Polytechnique de Paris, France
*bmiftari@uliege.be

Yet another modelling tool?

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

Mathematical formulation
This work focuses on block-decomposable problems that can be encoded by a hierarchical hypergraph \( G = (\bullet, \triangle) \), where
- \( \bullet \) is the set of nodes
- \( \triangle \) is the set of hyperedges linking these nodes.

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**

```python
#PARAMETERS
cost_invest = 128;
cost_op = 1;
irradiance = import "irradiance.csv";
max_capacity = 569.6;

#VARIABLES
internal: capacity;
external: electricity[T];

#CONSTRAINTS
electricity[t] <= irradiance[t] * capacity;
capacity <= max_capacity;
capacity >= 6;
electricity[t] >= 6;

#OBJECTIVES
min: cost_invest * capacity;
min: cost_op * electricity[t];
```

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

**NODE HOUSE**

```python
#PARAMETERS
demand = import "demand.csv";
energy_price = 2;

#NODE PV = import "PV.gboml" 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];
```

Benchmark

![Graph-Based Optimization Modelling Language (GBOML)](image)

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)