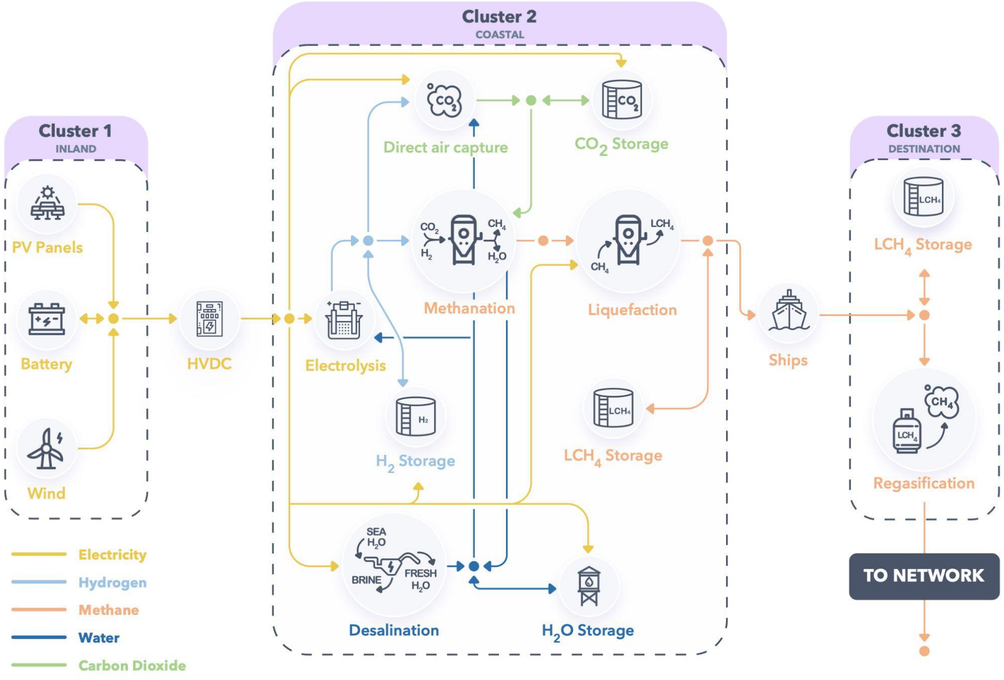
Graph-**B**ased **O**ptimization **M**odeling **L**anguage (GBOML)

TotalEnergies

May 19, 2022

Bardhyl Miftari
Mathias Berger
Damien Ernst

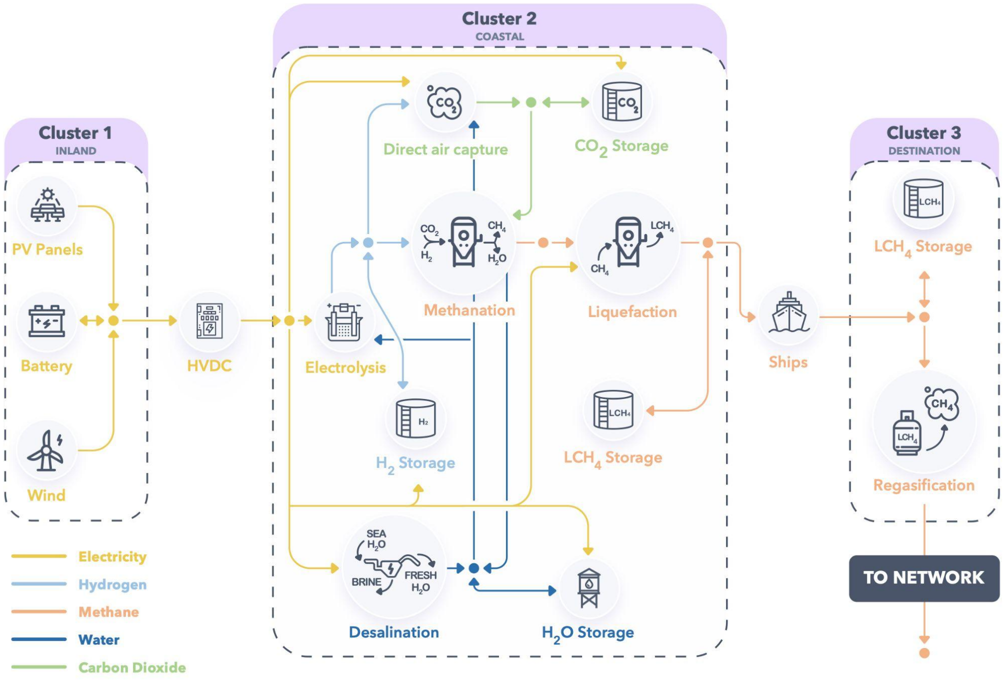
Many energy system planning and control problems can be formulated as mathematical programs



For example, we may want to *design* and *operate* a system producing synthetic fuel so as to minimise its cost

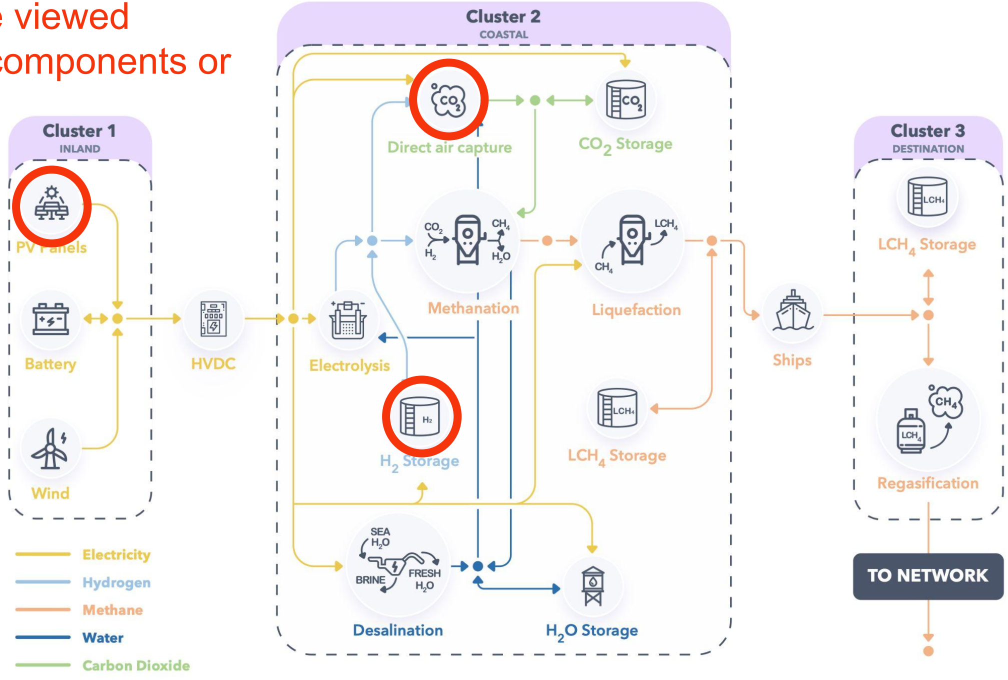
$$
\begin{aligned}
\min \quad & [\text{investment costs}] + [\text{operating costs}] \\
\text{s.t.} \quad & [\text{system design constraints}] \\
& [\text{operating constraints of subsystems}] \\
& [\text{coupling constraints between subsystems}] \\
& [\text{regulatory and environmental constraints}]
\end{aligned}
$$

Mathematical programs found in energy system planning and control applications have special structure



Notably, the underlying system is often a collection of subsystems whose operation must be optimized over time

Subsystems can be viewed as different model components or blocks



This structure can be represented via a hypergraph abstraction augmented with some concept of time-indexing

$$
\begin{aligned}
\min \quad & \textstyle\sum_{n \in \mathcal{N}} \left[f_0^n(X^n, Z^n) + \sum_{t \in \mathcal{T}} f^n(X^n, Z^n, t) \right] \\
\text{s.t.} \quad & h_k^n(X^n, Z^n, t) = 0, \ \forall t \in \mathcal{T}_k^n, \ k = 1, \dots K^n, \ \forall n \in \mathcal{N} \\
& g_k^n(X^n, Z^n, t) \le 0, \ \forall t \in \bar{\mathcal{T}}_k^n, \ k = 1, \dots \bar{K}^n, \ \forall n \in \mathcal{N} \\
& H^e(Z^e) = 0, \ \forall e \in \mathcal{E} \\
& G^e(Z^e) \le 0, \ \forall e \in \mathcal{E} \\
& X^n \in \mathcal{X}^n, Z^n \in \mathcal{Z}^n, \ \forall n \in \mathcal{N}.
\end{aligned}
$$

Sets of *nodes*, *hyperedges* and *time periods* define the hypergraph and time-indexed structure

set of nodes

set of time periods

$$
\begin{aligned}
\min \quad & \textstyle\sum_{n\in\mathcal{N}} \left[f_0^n(X^n, Z^n) + \sum_{t\in\mathcal{T}} f^n(X^n, Z^n, t) \right] \\
\text{s.t.} \quad & h_k^n(X^n, Z^n, t) = 0, \;\; \forall t \in \mathcal{T}_k^n, \;\; k = 1, \ldots K^n, \;\; \forall n \in \mathcal{N} \\
& g_k^n(X^n, Z^n, t) \leq 0, \;\; \forall t \in \bar{\mathcal{T}}_k^n, \;\; k = 1, \ldots \bar{K}^n, \;\; \forall n \in \mathcal{N} \\
& H^e(Z^e) = 0, \; \forall e \in \mathcal{E} \\
& G^e(Z^e) \leq 0, \; \forall e \in \mathcal{E} \\
& X^n \in \mathcal{X}^n, Z^n \in \mathcal{Z}^n, \; \forall n \in \mathcal{N}.
\end{aligned}
$$

set of hyperedges

We distinguish between *internal* variables belonging to nodes and *coupling* variables used to link nodes

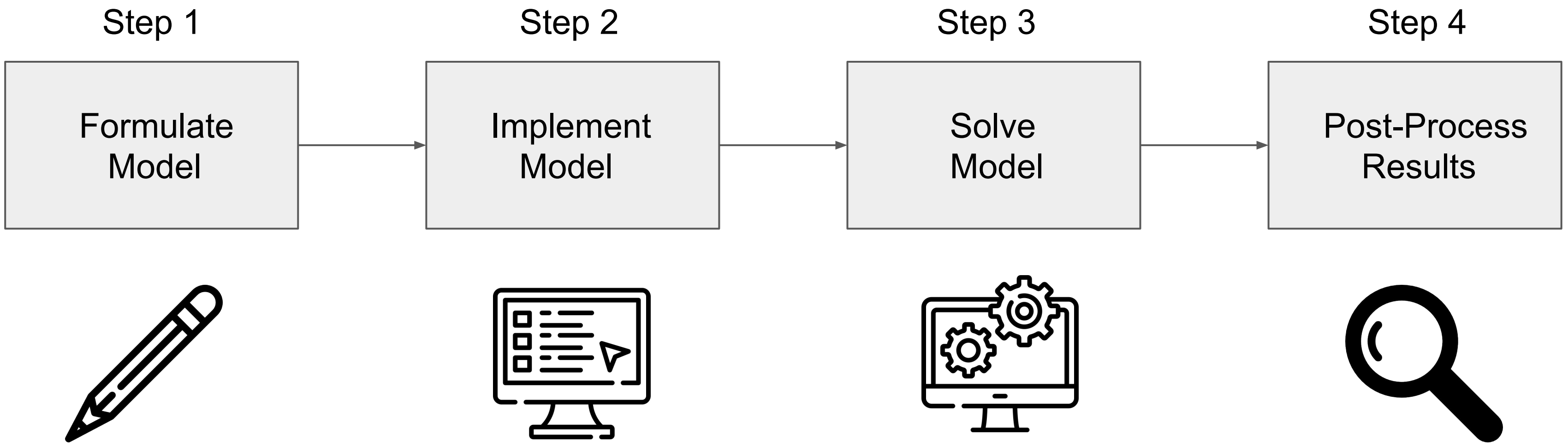
internal variables coupling variables

$$
\begin{aligned}
\min \quad & \textstyle\sum_{n \in \mathcal{N}} \left[f_0^n(X^n, Z^n) + \sum_{t \in \mathcal{T}} f^n(X^n, Z^n, t) \right] \\
\text{s.t.} \quad & h_k^n(X^n, Z^n, t) = 0,\ \forall t \in \mathcal{T}_k^n,\ k = 1, \ldots K^n,\ \forall n \in \mathcal{N} \\
& g_k^n(X^n, Z^n, t) \leq 0,\ \forall t \in \bar{\mathcal{T}}_k^n,\ k = 1, \ldots \bar{K}^n,\ \forall n \in \mathcal{N} \\
& H^e(Z^e) = 0,\ \forall e \in \mathcal{E} \\
& G^e(Z^e) \leq 0,\ \forall e \in \mathcal{E} \\
& X^n \in \mathcal{X}^n, Z^n \in \mathcal{Z}^n,\ \forall n \in \mathcal{N}.
\end{aligned}
$$

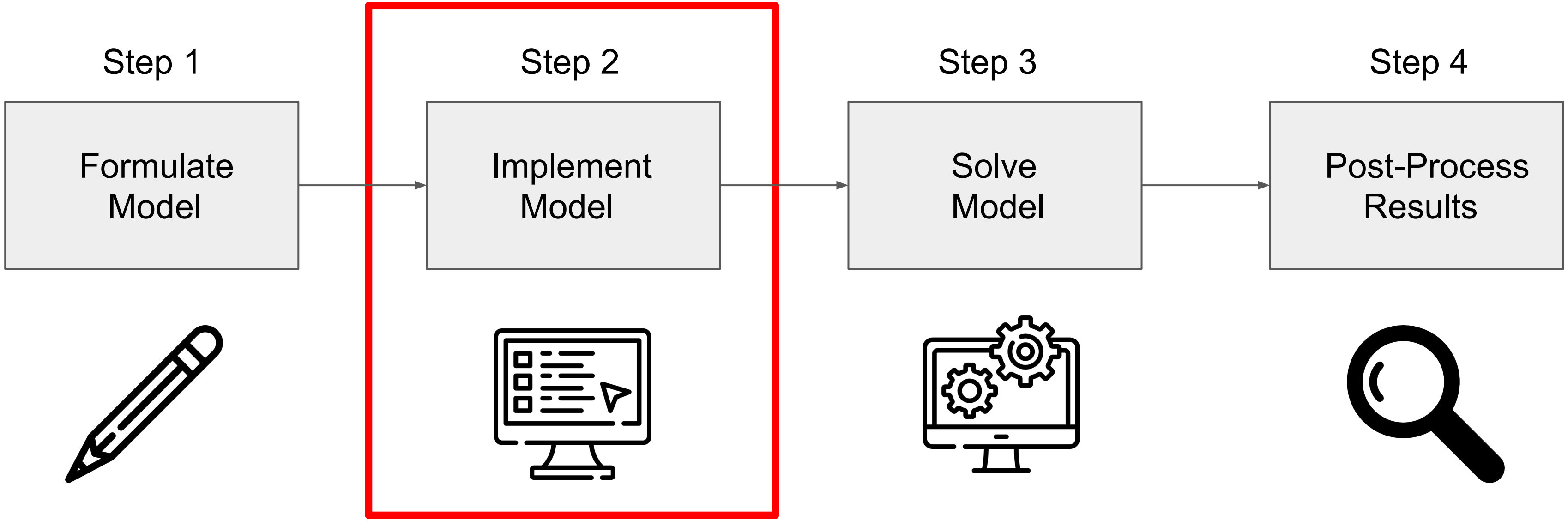
Each node has its own objective and constraints while hyperedges implement coupling constraints

$$
\begin{aligned}
\min\quad & \sum_{n\in\mathcal{N}}\left[\overbrace{f_0^n(X^n,Z^n)+\sum_{t\in\mathcal{T}} f^n(X^n,Z^n,t)}^{\text{node objective}}\right]\\
\text{s.t.}\quad & \left.\begin{array}{l} h_k^n(X^n,Z^n,t)=0,\ \forall t\in\mathcal{T}_k^n,\ k=1,\dots K^n,\ \forall n\in\mathcal{N}\\ g_k^n(X^n,Z^n,t)\le 0,\ \forall t\in\bar{\mathcal{T}}_k^n,\ k=1,\dots \bar{K}^n,\ \forall n\in\mathcal{N}\end{array}\right\}\text{node constraints}\\
& \left.\begin{array}{l} H^e(Z^e)=0,\ \forall e\in\mathcal{E}\\ G^e(Z^e)\le 0,\ \forall e\in\mathcal{E}\end{array}\right\}\text{coupling constraints}\\
& X^n\in\mathcal{X}^n,\ Z^n\in\mathcal{Z}^n,\ \forall n\in\mathcal{N}.
\end{aligned}
$$

Working with optimization models involves at least four basic steps



We will focus on the second step: model encoding and implementation



Two classes of tools are available to implement models

1. Algebraic Modeling Languages (AMLs):

- **Formulation** close to **mathematical notation** (e.g., index-based notation)
- Very **expressive** (e.g., can represent any mixed-integer nonlinear program)
- Often interface with **multiple solvers**
- Sometimes **open source**
- Examples :

Two classes of tools are available to implement models

2. Object-Oriented Modeling Environments (OOMEs):

- Focus on **one** particular **application** (e.g., generation expansion planning)
- Usually make use of **predefined components** that can be "imported"
- Typically have advanced **data processing** capabilities tailored to application
- Often **open source**
- Examples :

Each approach has drawbacks

AMLs typically fail to **expose** or **exploit** block **structure**, although this may be used to:

- simplify model encoding
- enable model re-use
- speed up model generation
- facilitate the use of structure-exploiting algorithms

OOMEs, for their part:

- Lack **expressiveness**
- Often **cumbersome** to add **new components**
- Usually **rely** on **AMLs** and inherit any **shortcomings**

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone
- any mixed-integer linear program can be represented

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone
- any mixed-integer linear program can be represented
- hierarchical block structure can be exposed and exploited

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone
- any mixed-integer linear program can be represented
- hierarchical block structure can be exposed and exploited
- syntax is close to mathematical notation

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone
- any mixed-integer linear program can be represented
- hierarchical block structure can be exposed and exploited
- syntax is close to mathematical notation
- time-indexed models can be encoded easily

GBOML combines the strengths of AMLs and OOMEs

- open-source and stand-alone
- any mixed-integer linear program can be represented
- hierarchical block structure can be exposed and exploited
- syntax is close to mathematical notation
- time-indexed models can be encoded easily
- re-using and combining model components is straightforward

GBOML combines the strengths of AMLs and OOMEs

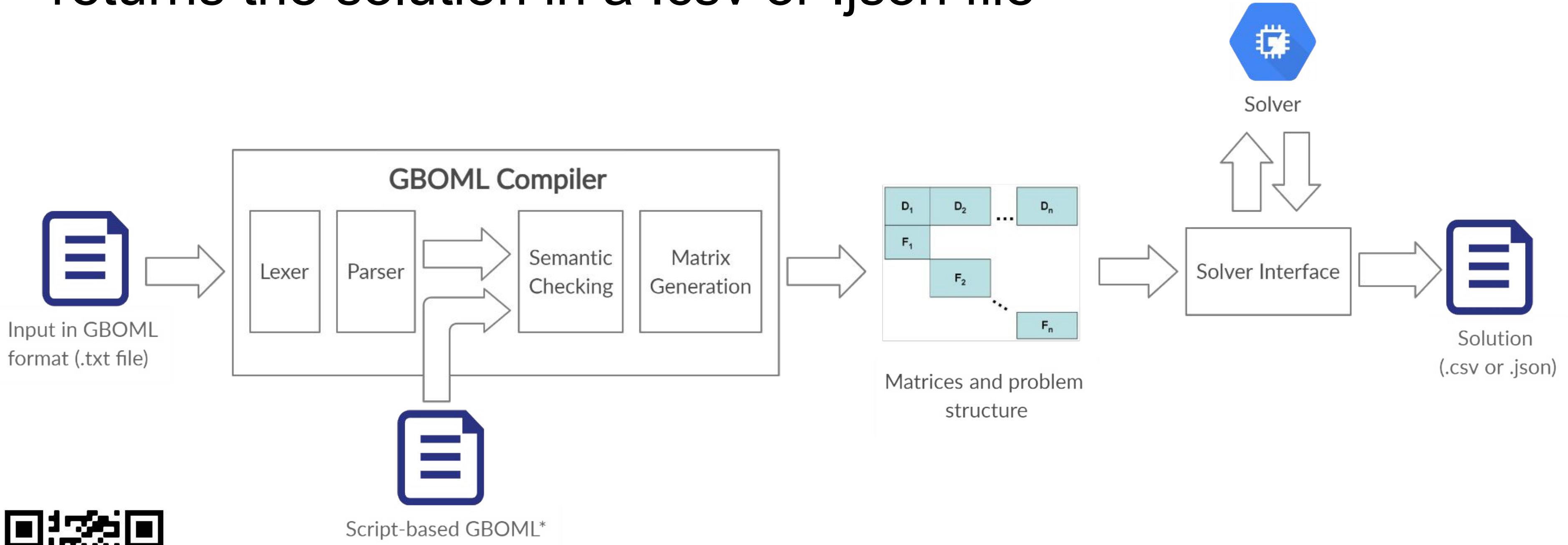
- open-source and stand-alone
- any mixed-integer linear program can be represented
- hierarchical block structure can be exposed and exploited
- syntax is close to mathematical notation
- time-indexed models can be encoded easily
- re-using and combining model components is straightforward
- interfaces with various solvers are available

GBOML is a modeling language with its own parser

- Software developed in Python:
  - Has very **few dependencies** (PLY, NumPy, SciPy)
  - Provides two methods to **encode** models (**file** and Python **API**)
  - Interfaces with **several solvers** (Gurobi, CPLEX, Xpress, Cbc/Clp, HiGHS and DSP)
  - Produces plain **.csv** or structured **.json** outputs

- Model structure is exploited on multiple levels:
  - Model **encoding** via dedicated language constructs
  - Model **generation** via parallelism and multiprocessing
  - **Solving** via structure-exploiting solvers such as DSP

GBOML takes inputs from a file or Python script and returns the solution in a .csv or .json file



A Jupyter notebook has been prepared to illustrate how GBOML can be used

References

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

[2] Miftari et al., (2022). GBOML: Graph-Based Optimization Modeling Language. Journal of Open Source Software, 7(72), 4158, https://doi.org/10.21105/joss.04158

[3] Miftari et al., (2022). GBOML repository: https://gitlab.uliege.be/smart_grids/public/gboml

[4] Freepik Icons (used on slides 10-11): https://www.flaticon.com/authors/freepik