GBOML: A modelling tool for structured MILPs

TB-06: Software for Optimization 2: Modelling

Bardhyl MIFTARI, Mathias BERGER, Guillaume DERVAL and Damien ERNST
University of Liège Belgium
11 JULY 2023

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

IFORS International Federation of Operational Research Societies

LIÈGE université

ICHIO INSTITUTO CHILENO DE INVESTIGACIÓN OPERATIVA

alio asociación latino iberoamericana de investigación operativa

Universidad de Chile

Pontificia Universidad Católica de Chile

ISCI Instituto Sistemas Complejos de Ingeniería

Examples of structured MILPs

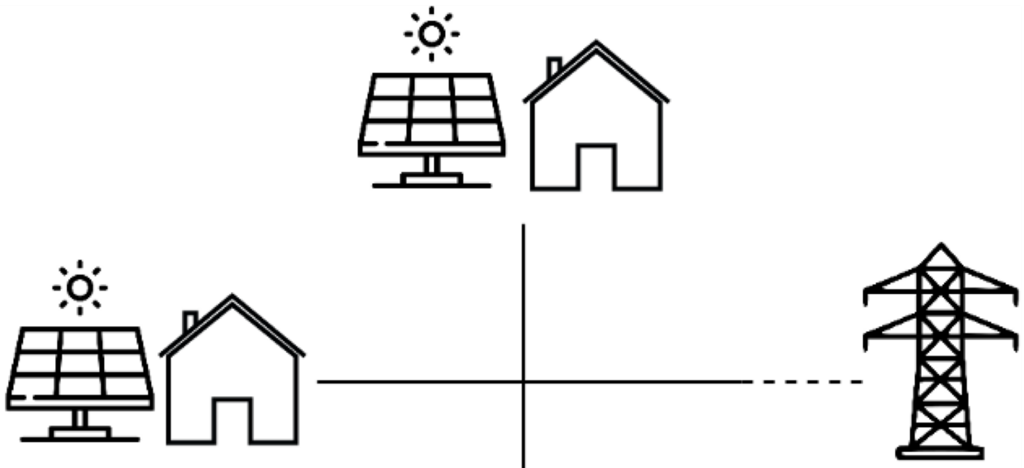


Figure 1: Renewable energy community

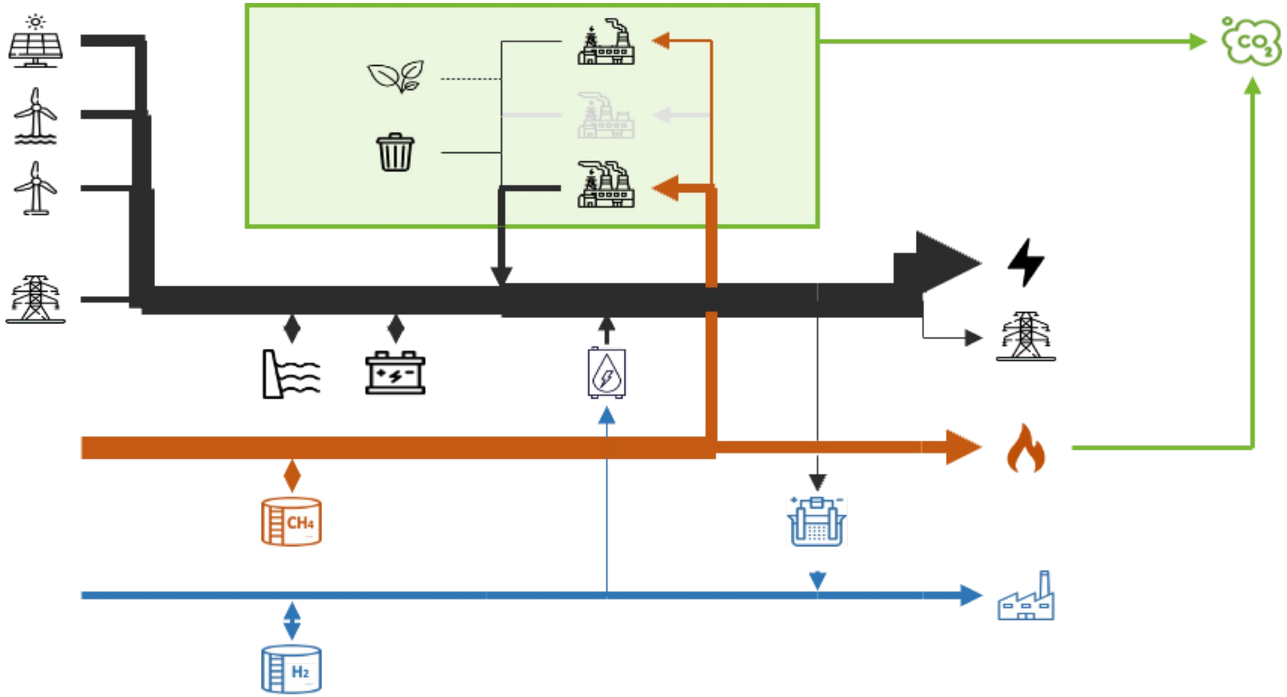


Figure 2: Belgian energy model

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10 - 14 · SANTIAGO, CHILE

Examples of structured MILPs

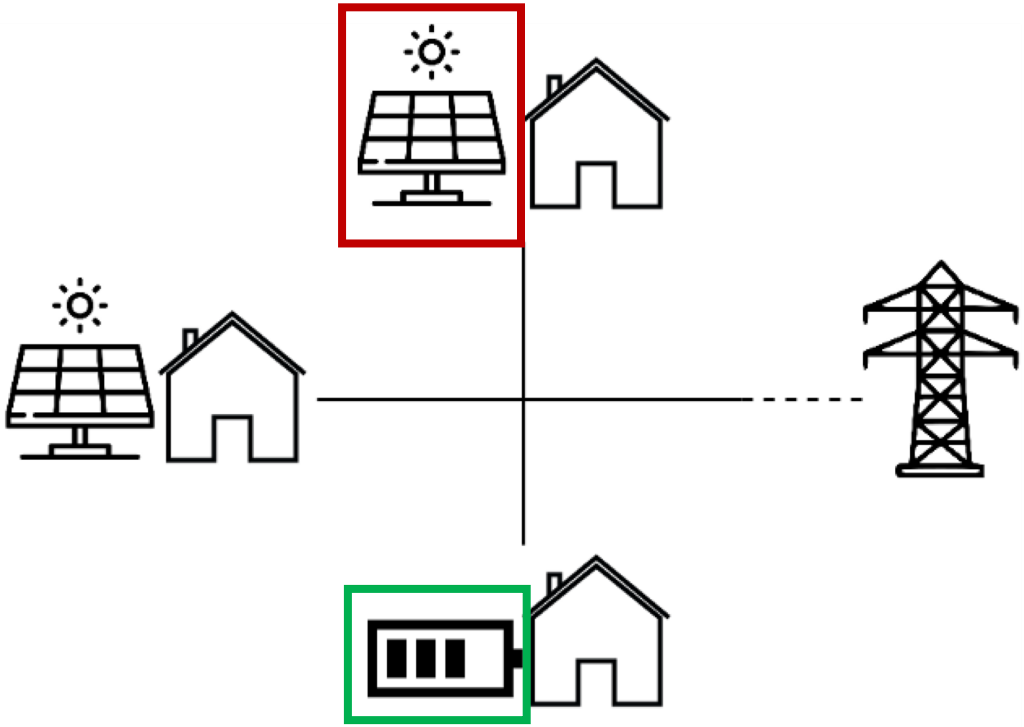


Figure 1: Renewable energy community

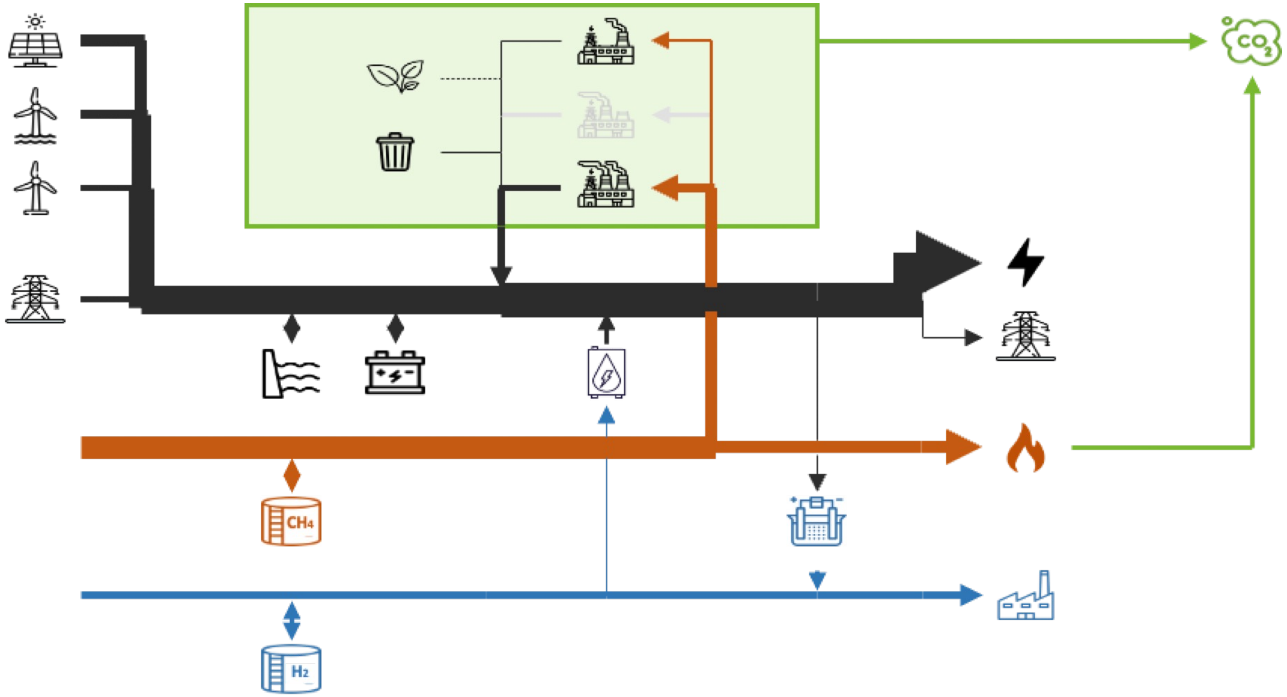


Figure 2: Belgian energy model

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Examples of structured MILPs

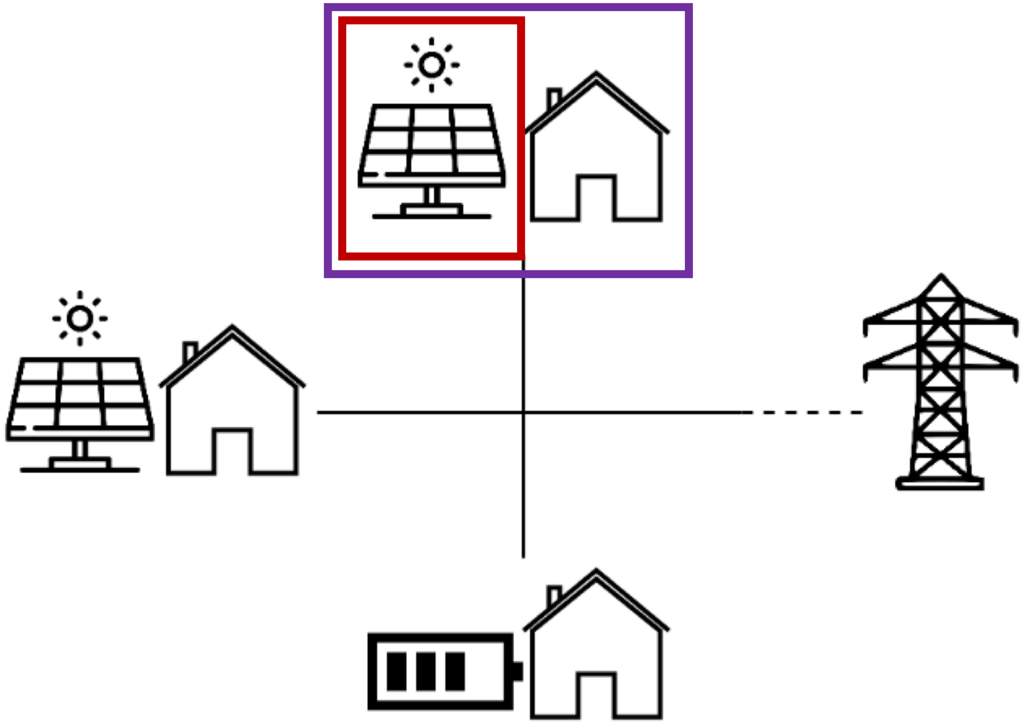


Figure 1: Renewable energy community

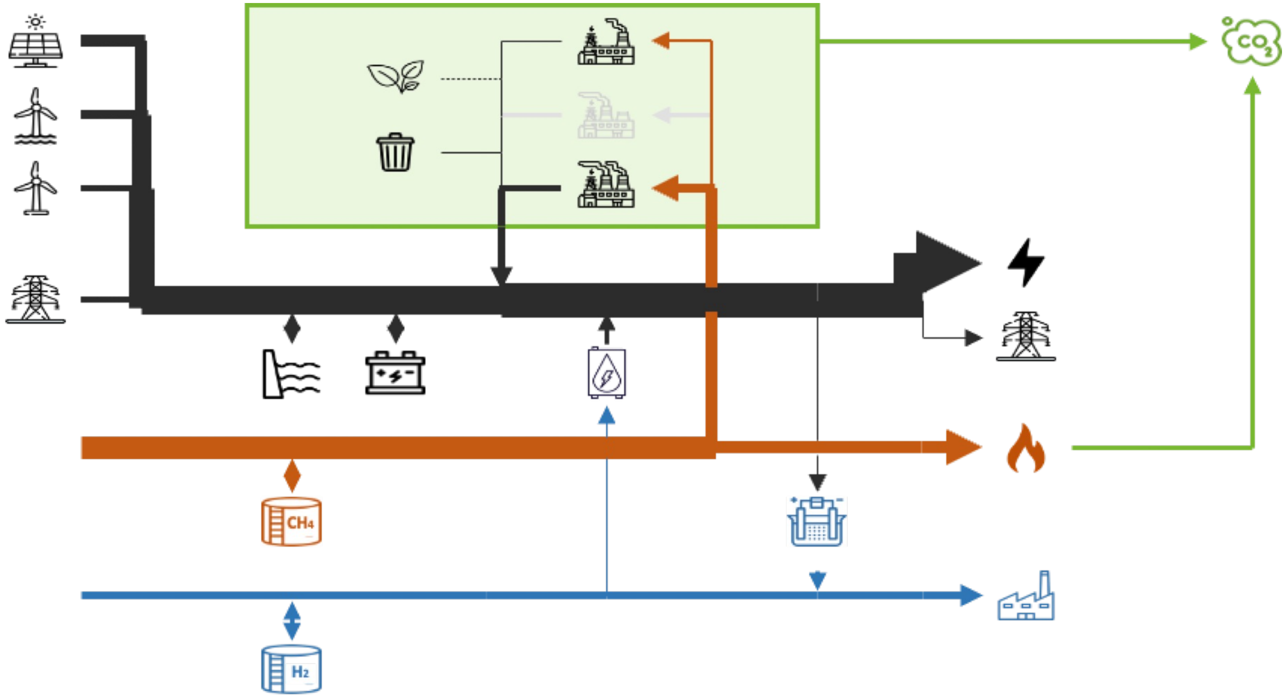


Figure 2: Belgian energy model

Examples of structured MILPs

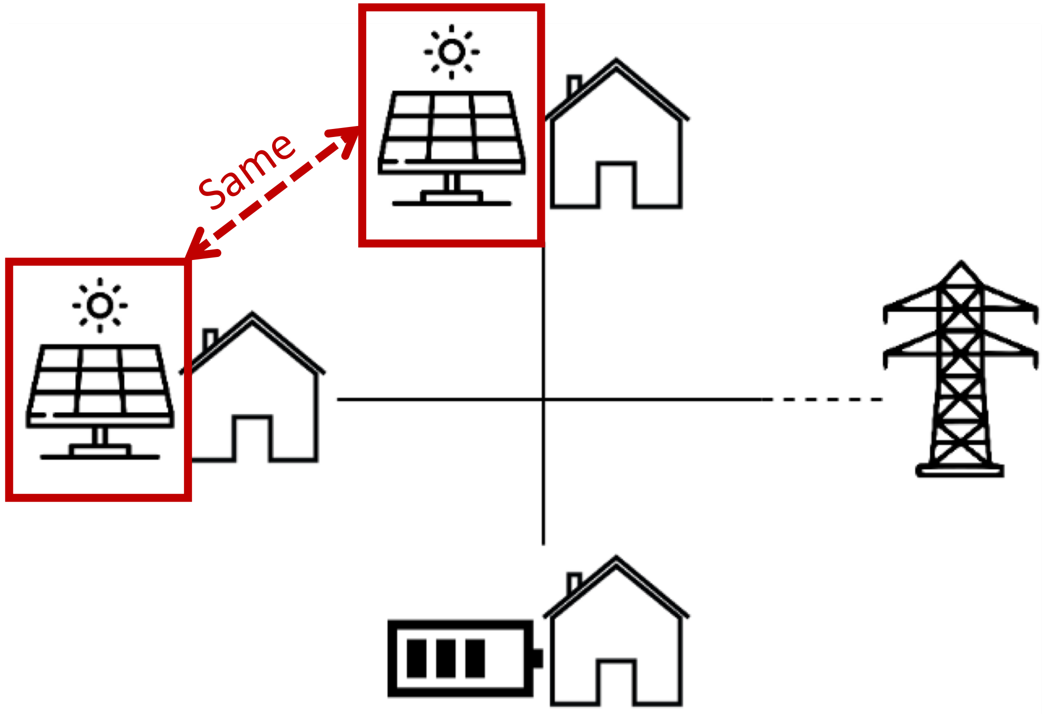


Figure 1: Renewable energy community

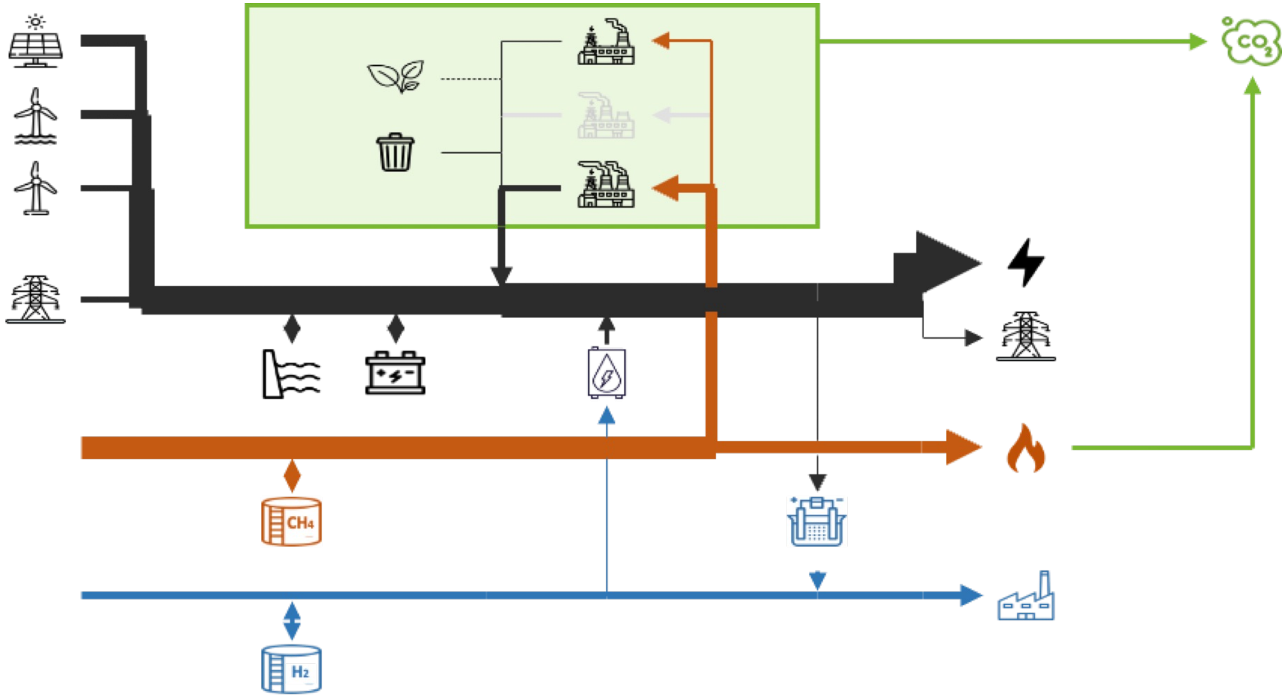


Figure 2: Belgian energy model

Examples of structured MILPs

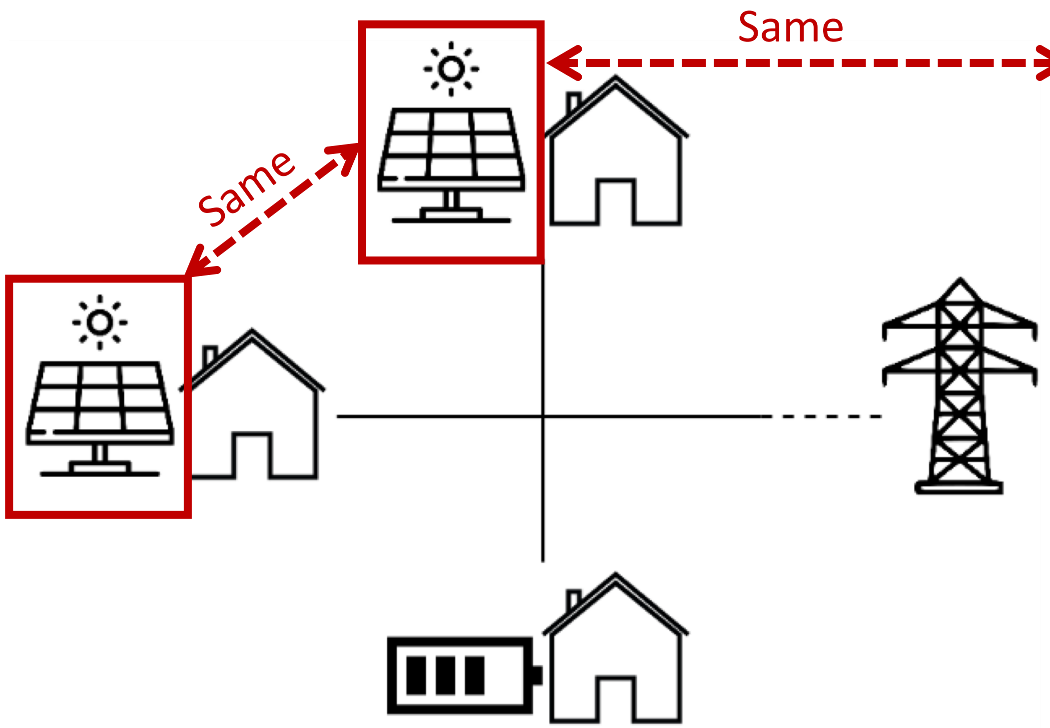


Figure 1: Renewable energy community

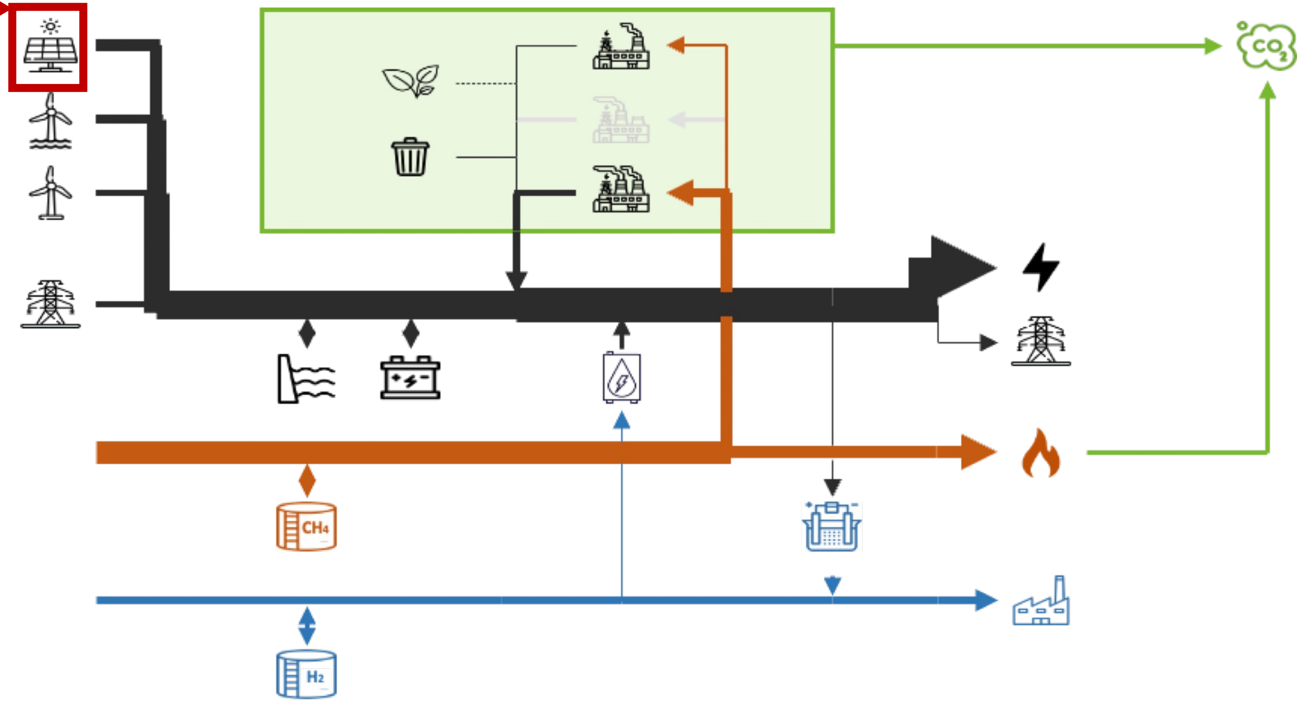
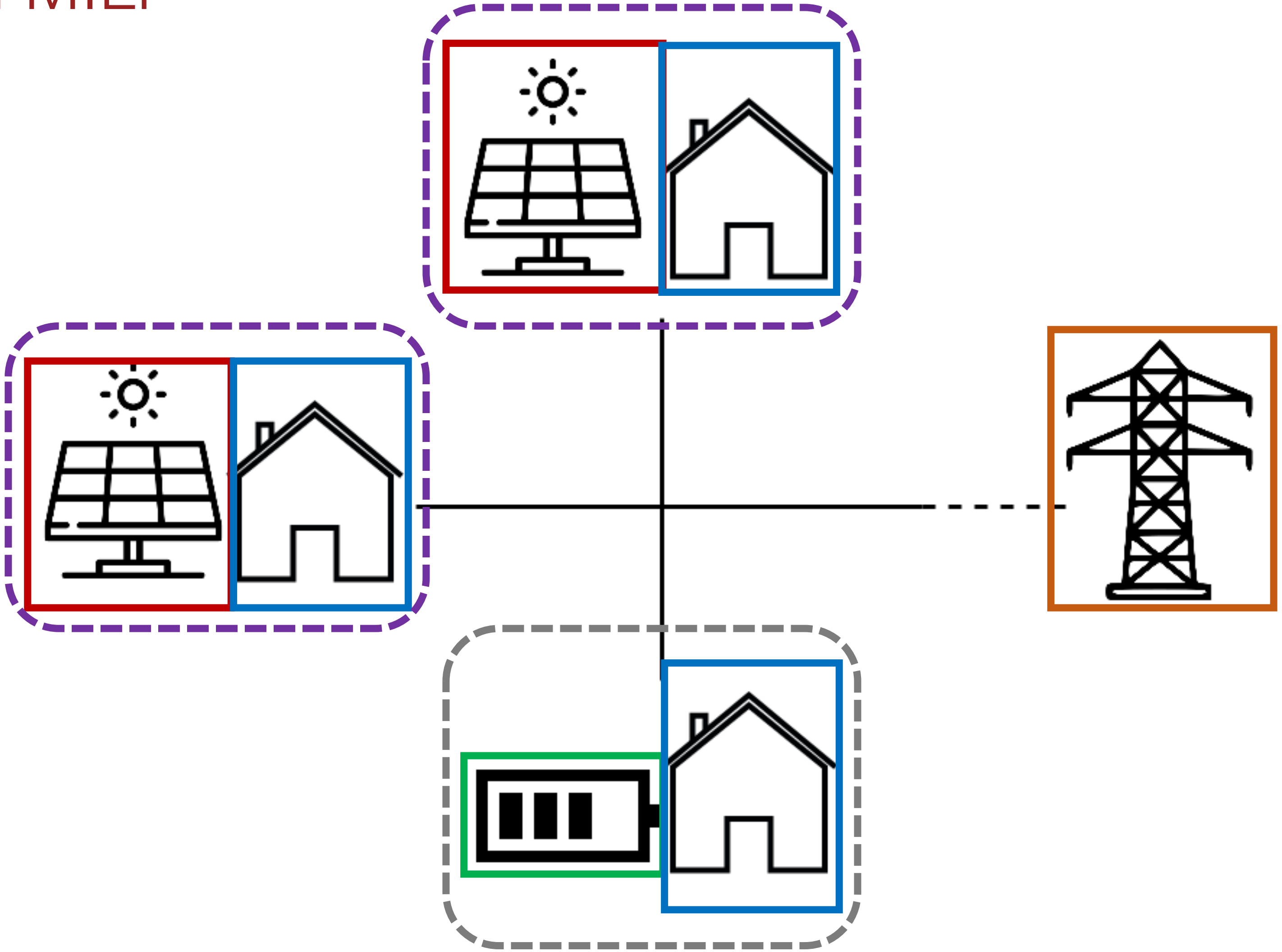


Figure 2: Belgian energy model

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Let us formalize these problems

A structured MILP



PV Panels
House
Grid
Battery
Prosumer
Bat-consumer

Figure 1: Renewable energy community

A structured MILP

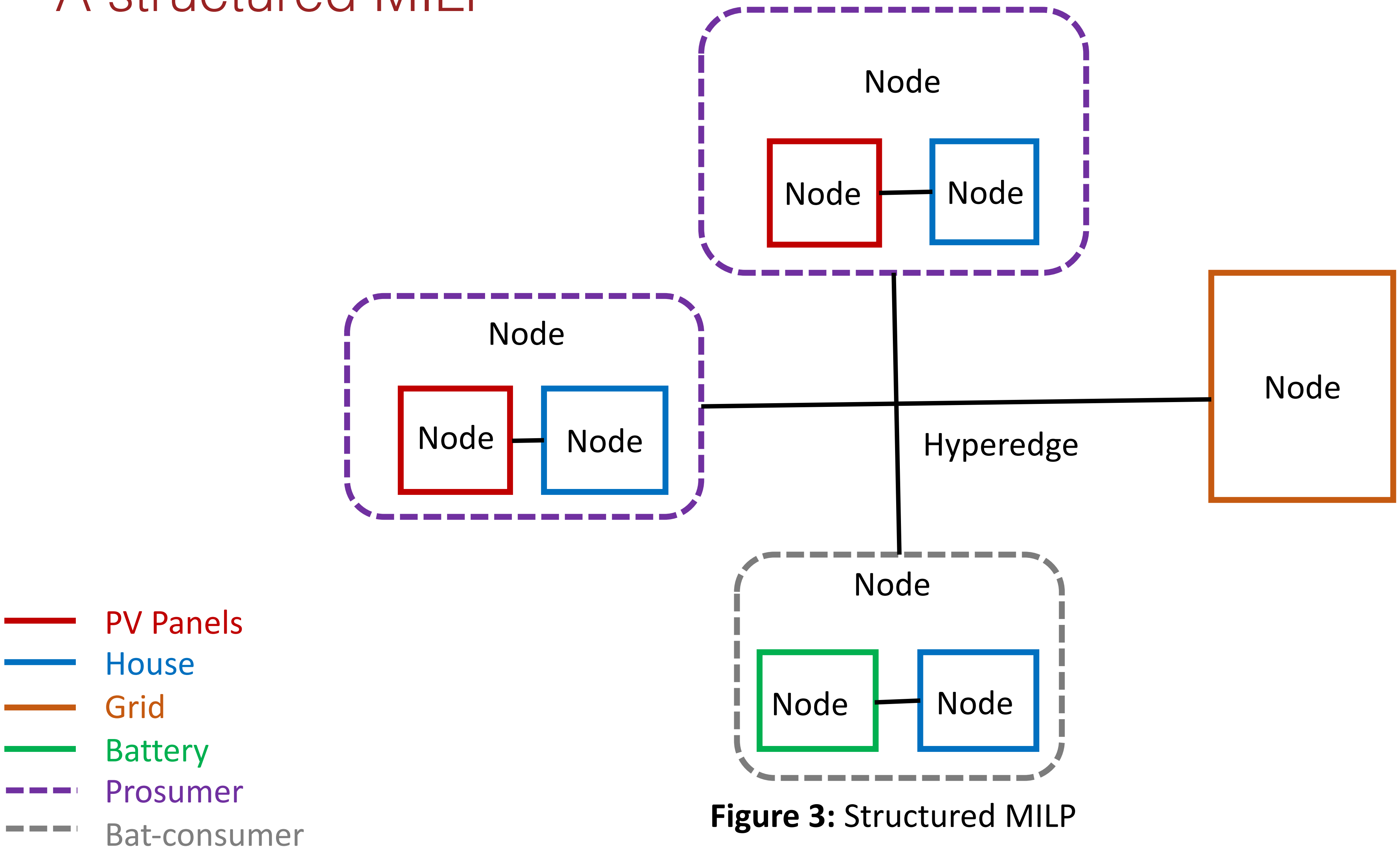


Figure 3: Structured MILP

Structured MILPs abstraction

Hierarchical hypergraph made of

- **Nodes**
- **Hyperedges**

Each **node** is itself made of

- **Parameters**
- **Variables**
- **Constraints**
- **Objectives**
- **Hierarchical hypergraph**

Each **hyperedge** is made of

- **Parameters**
- **Constraints**

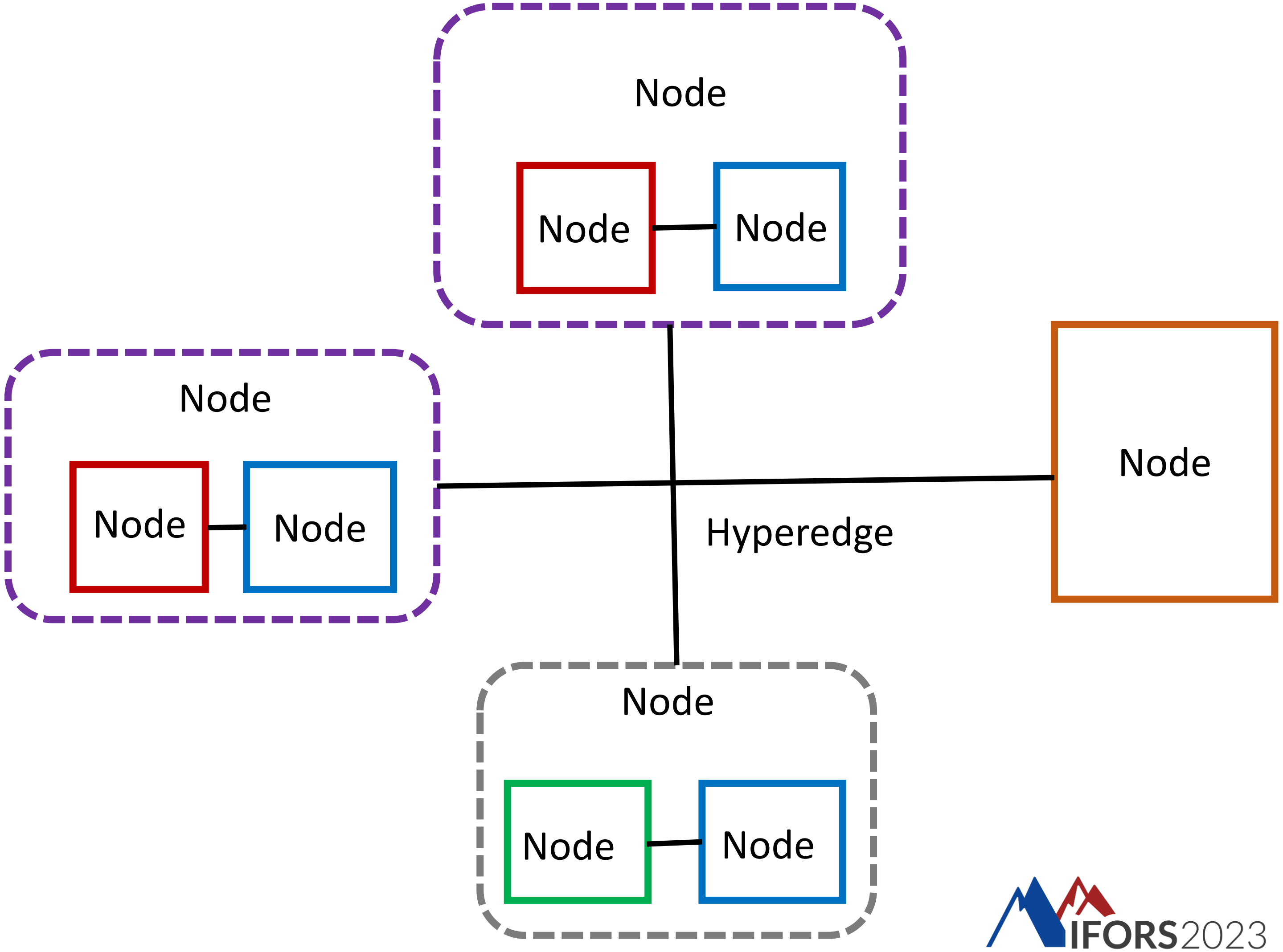


Figure 3: Structured MILP

Structured MILPs formalization

Hierarchical hypergraph $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ made of

- A set of **nodes** $\mathcal{N}_n$
- A set of **hyperedges** $\mathcal{E}_n$

Each **node** $< v_n^{ext}, v_n^{int}, \boldsymbol{G_n}, \boldsymbol{H_n}, G_n, \boldsymbol{O_n} >$ is itself made of

- **External** variables v_n^{ext}
- **Internal** variables v_n^{int}

$\left.\begin{matrix} v_n^{ext} \\ v_n^{int} \end{matrix}\right] v_n$

- **Constraints** matrices $\boldsymbol{G_n}$ *and* $\boldsymbol{H_n}$
- **Objectives** matrix $\boldsymbol{O_n}$
- **Hierarchical hypergraph** G_n

Each **hyperedge** $< \mathcal{N}_e, \boldsymbol{G_e}, \boldsymbol{H_e} >$ is made of

- A set of nodes $\mathcal{N}_e$
- **Constraints** matrices $\boldsymbol{G_e}$ *and* $\boldsymbol{H_e}$

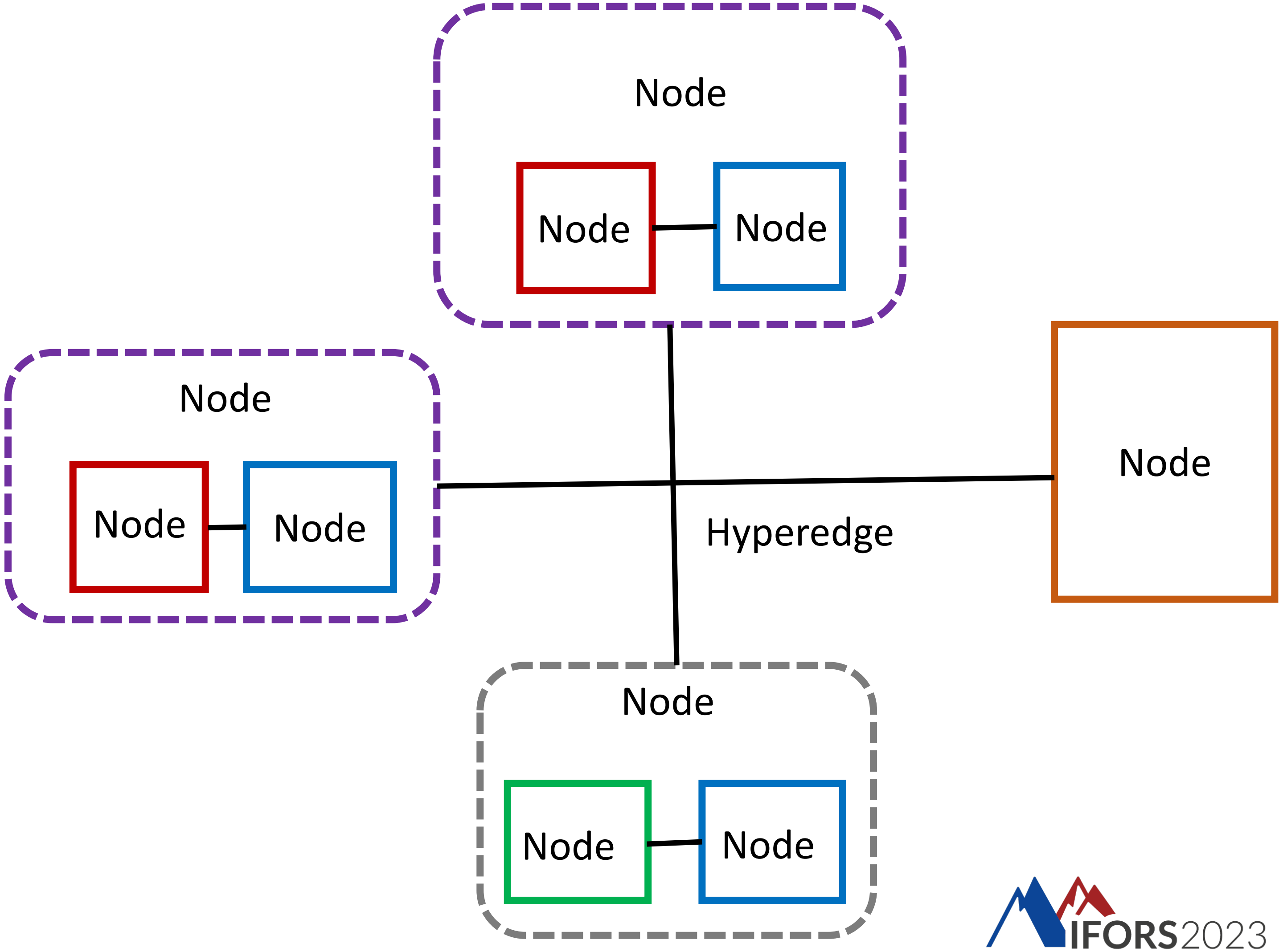


Figure 3: Structured MILP

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Structured MILPs formalization

Hierarchical hypergraph $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ made of

- A set of **nodes** $\mathcal{N}_n$
- A set of **hyperedges** $\mathcal{E}_n$

Each **node** $< \boldsymbol{v_n^{ext}}, \boldsymbol{v_n^{int}}, \boldsymbol{G_n}, \boldsymbol{H_n}, G_n, \boldsymbol{O_n} >$ is itself made of

- **External** variables $\boldsymbol{v_n^{ext}}$
- **Internal** variables $\boldsymbol{v_n^{int}}$

$\left.\begin{matrix} \boldsymbol{v_n^{ext}} \\ \boldsymbol{v_n^{int}} \end{matrix}\right] \boldsymbol{v_n}$

- **Constraints** matrices $\boldsymbol{G_n}$ *and* $\boldsymbol{H_n}$
- **Objectives** matrix $\boldsymbol{O_n}$
- **Hierarchical hypergraph** G_n

Each **hyperedge** $< \mathcal{N}_e, \boldsymbol{G_e}, \boldsymbol{H_e} >$ is made of

- A set of nodes $\mathcal{N}_e$
- **Constraints** matrices $\boldsymbol{G_e}$ *and* $\boldsymbol{H_e}$

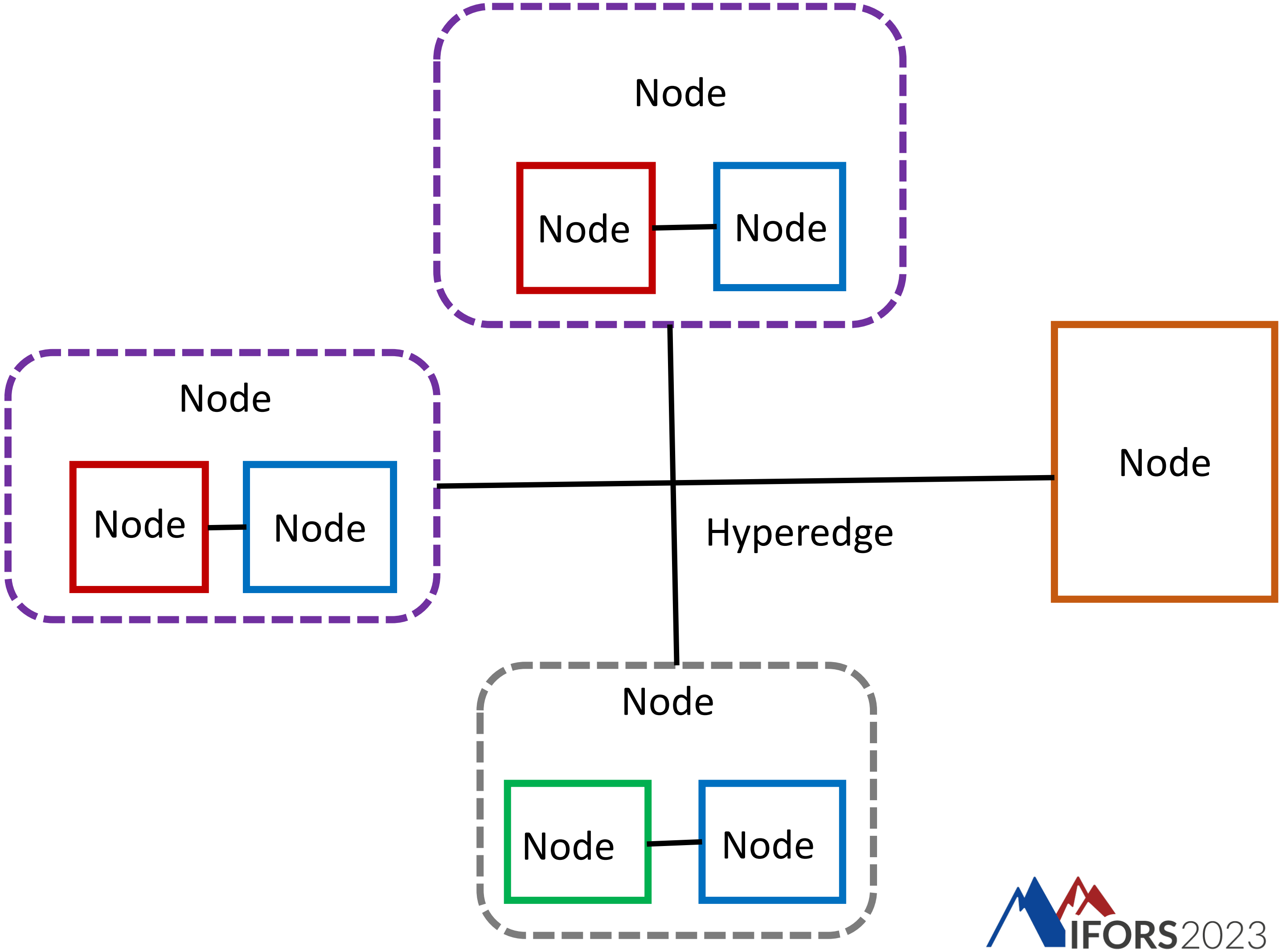


Figure 3: Structured MILP

Structured MILPs formalization

Hierarchical hypergraph $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ made of

- A set of **nodes** $\mathcal{N}_n$
- A set of **hyperedges** $\mathcal{E}_n$

Each **node** $< \boldsymbol{v_n^{ext}}, \boldsymbol{v_n^{int}}, \boldsymbol{G_n}, \boldsymbol{H_n}, G_n, \boldsymbol{O_n} >$ is itself made of

- **External** variables $\boldsymbol{v_n^{ext}}$
- **Internal** variables $\boldsymbol{v_n^{int}}$

$$\left.\begin{array}{l} \boldsymbol{v_n^{ext}} \\ \boldsymbol{v_n^{int}} \end{array}\right\} \boldsymbol{v_n}$$

- **Constraints** matrices $\boldsymbol{G_n}$ *and* $\boldsymbol{H_n}$
- **Objectives** matrix $\boldsymbol{O_n}$
- **Hierarchical hypergraph** G_n

Each **hyperedge** $< \mathcal{N}_e, \boldsymbol{G_e}, \boldsymbol{H_e} >$ is made of

- A set of nodes $\mathcal{N}_e$
- **Constraints** matrices $\boldsymbol{G_e}$ *and* $\boldsymbol{H_e}$

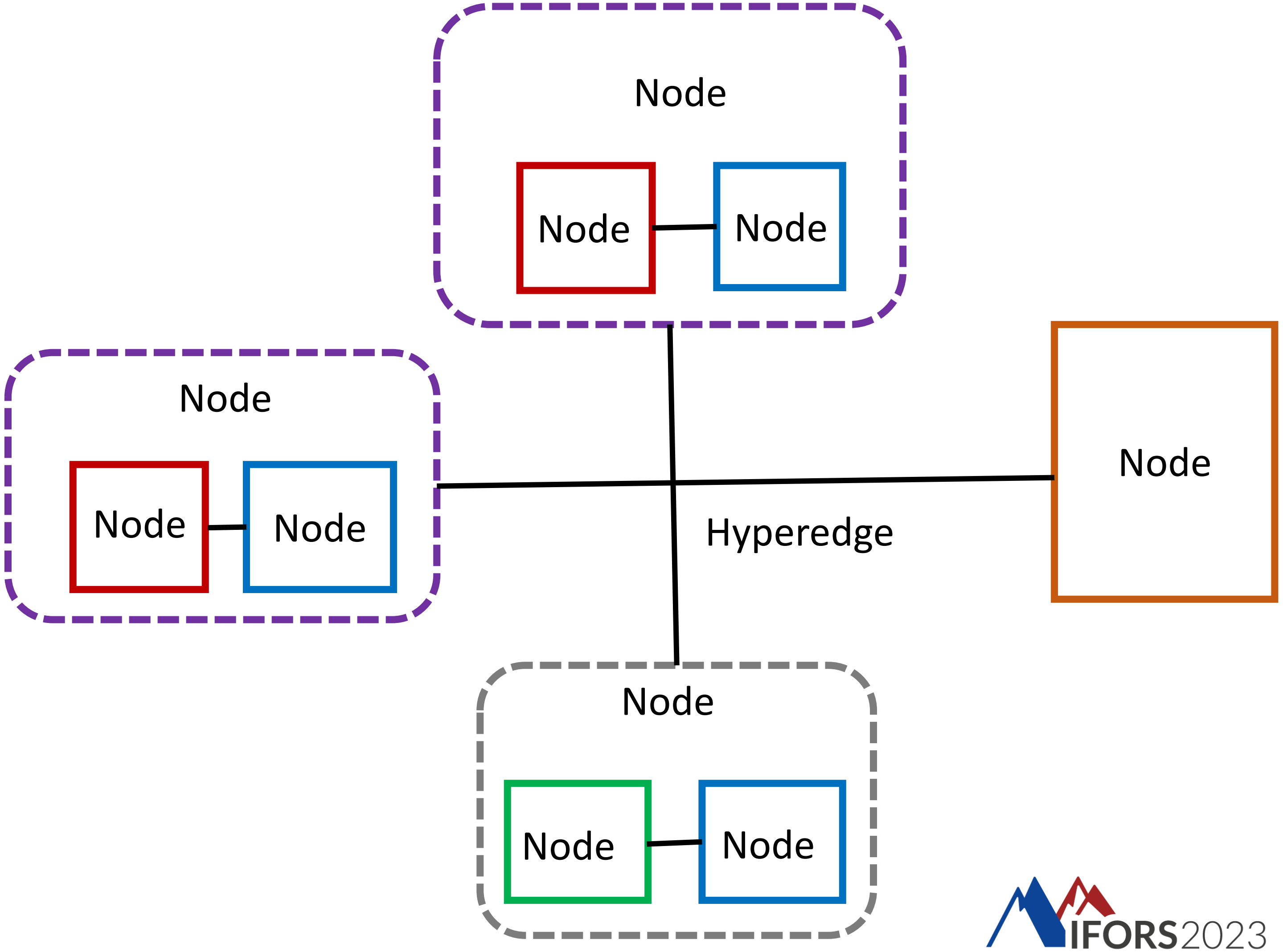


Figure 3: Structured MILP

Structured MILPs formalization

We have a **hierarchical hypergraph** $G_g = (\mathcal{N}_g, \mathcal{E}_g)$

For each node, we have

$$\begin{aligned} \min \mathbf{1}^{1\times\sigma_n}\ \boldsymbol{O_n}[1\ \boldsymbol{v_n}]^T & \\ \text{s.t.}\ \boldsymbol{G_n}[1\ \boldsymbol{v_n}]^T & \le \mathbf{0} \\ \boldsymbol{H_n}[1\ \boldsymbol{v_n}]^T & = \mathbf{0} \end{aligned}$$

For each hyperedge, we must respect the constraints

$$\begin{aligned} \boldsymbol{G_e}[1\ \boldsymbol{v_e}]^T & \le \mathbf{0} \\ \boldsymbol{H_e}[1\ \boldsymbol{v_e}]^T & = \mathbf{0} \end{aligned}$$

And so on recursively for all sub-hypergraphs.

Structured MILPs formalization

Set of nodes

We have a **hierarchical hypergraph** $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ Set of hyperedges

For each node, we have

$$\begin{aligned} \min\ & \mathbf{1}^{1\times\sigma_n}\ \boldsymbol{O}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^T \\ \text{s.t.}\ & \boldsymbol{G}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^T \le \mathbf{0} \\ & \boldsymbol{H}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^T = \mathbf{0} \end{aligned}$$

For each hyperedge, we must respect the constraints

$$\begin{aligned} \boldsymbol{G}_{\boldsymbol{e}}[1\ \boldsymbol{v}_{\boldsymbol{e}}]^T &\le \mathbf{0} \\ \boldsymbol{H}_{\boldsymbol{e}}[1\ \boldsymbol{v}_{\boldsymbol{e}}]^T &= \mathbf{0} \end{aligned}$$

And so on recursively for all sub-hypergraphs.

Structured MILPs formalization

Set of nodes

We have a **hierarchical hypergraph** $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ Set of hyperedges

For each node, we have

Objective matrix

$$\min \mathbf{1}^{1\times\sigma_n} \; \boldsymbol{O_n}[1\; \boldsymbol{v_n}]^T$$

Inequality constraints

Equality constraints

$$\text{s.t.}\; \boldsymbol{G_n}[1\; \boldsymbol{v_n}]^T \leq \mathbf{0}$$
$$\boldsymbol{H_n}[1\; \boldsymbol{v_n}]^T = \mathbf{0}$$

Node variables

For each hyperedge, we must respect the constraints

$$\boldsymbol{G_e}[1\; \boldsymbol{v_e}]^T \leq \mathbf{0}$$
$$\boldsymbol{H_e}[1\; \boldsymbol{v_e}]^T = \mathbf{0}$$

And so on recursively for all sub-hypergraphs.

Structured MILPs formalization

Set of nodes

We have a **hierarchical hypergraph** $G_g = (\mathcal{N}_g, \mathcal{E}_g)$ Set of hyperedges

For each node, we have

Objective matrix

$$\min \mathbf{1}^{1\times\sigma_n} \ \boldsymbol{O_n}[1\ \boldsymbol{v_n}]^T$$
$$\text{s.t. } \boldsymbol{G_n}[1\ \boldsymbol{v_n}]^T \leq \mathbf{0}$$
$$\boldsymbol{H_n}[1\ \boldsymbol{v_n}]^T = \mathbf{0}$$

Inequality constraints
Equality constraints

Node variables

For each hyperedge, we must respect the constraints

$$\boldsymbol{G_e}[1\ \boldsymbol{v_e}]^T \leq \mathbf{0}$$
$$\boldsymbol{H_e}[1\ \boldsymbol{v_e}]^T = \mathbf{0}$$

Inequality constraints
Equality constraints

External variables of certain nodes

And so on recursively for all sub-hypergraphs.

Structured MILPs formalization

Let us define three recursive functions :

- the function f that takes a set of nodes $\mathcal{N}$ as input and returns the sum of the objectives of the nodes and their subnodes recursively,

$$f(\mathcal{N}) = \sum_{n \in \mathcal{N}} \left(\mathbf{1}^{1\times\sigma_n}\, \boldsymbol{O}_n [1\ \boldsymbol{v}_n]^T + f(\mathcal{N}_n) \right).$$

- the Boolean-valued function g that takes a hypergraph $G = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$g(G) = [\boldsymbol{G}_e [1\ \boldsymbol{v}_e]^T \leq 0\ \forall e \in \mathcal{E}] \wedge \left[\left(\boldsymbol{G}_n [1\ \boldsymbol{v}_n]^T \leq 0 \wedge g(G_n) \right) \forall n \in \mathcal{N} \right].$$

- the Boolean-valued function h that takes a hypergraph $G = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$h(G) = [\boldsymbol{H}_e [1\ \boldsymbol{v}_e]^T = 0\ \forall e \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H}_n [1\ \boldsymbol{v}_n]^T = 0 \wedge h(G_n) \right) \forall n \in \mathcal{N} \right].$$

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Structured MILPs formalization

Let us define three recursive functions :

- the function f that takes a set of nodes $\mathcal{N}$ as input and returns the sum of the objectives of the nodes and their subnodes recursively,

$$f(\mathcal{N}) = \sum_{n \in \mathcal{N}} \left(\mathbf{1}^{\mathbf{1}\times\boldsymbol{\sigma_n}} \, \boldsymbol{O_n} [1 \; \boldsymbol{v_n}]^T + \boxed{f(\mathcal{N}_n)} \right).$$

Applied recursively on its sub-nodes

- the Boolean-valued function g that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{g}(\mathrm{G}) = [\boldsymbol{G_e}[1 \; \boldsymbol{v_e}]^T \leq 0 \; \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{G_n}[1 \; \boldsymbol{v_n}]^T \leq 0 \wedge g(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

- the Boolean-valued function h that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{h}(\mathrm{G}) = [\boldsymbol{H_e}[1 \; \boldsymbol{v_e}]^{\mathrm{T}} = 0 \; \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H_n}[1 \; \boldsymbol{v_n}]^{\mathrm{T}} = 0 \wedge \mathrm{h}(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

Structured MILPs formalization

Let us define three recursive functions :

- the function f that takes a set of nodes $\mathcal{N}$ as input and returns the sum of the objectives of the nodes and their subnodes recursively,

$$f(\mathcal{N}) = \sum_{n \in \mathcal{N}} \left(\mathbf{1}^{1\times\sigma_n} \, \boldsymbol{O}_{\boldsymbol{n}}[1 \; \boldsymbol{v}_{\boldsymbol{n}}]^T + f(\mathcal{N}_n) \right).$$

- the Boolean-valued function g that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{g}(\mathrm{G}) = [\boldsymbol{G}_{\boldsymbol{e}}[1 \; \boldsymbol{v}_{\boldsymbol{e}}]^T \leq 0 \; \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{G}_{\boldsymbol{n}}[1 \; \boldsymbol{v}_{\boldsymbol{n}}]^T \leq 0 \wedge g(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

- the Boolean-valued function h that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{h}(\mathrm{G}) = [\boldsymbol{H}_e[1 \; \boldsymbol{v}_e]^\mathrm{T} = 0 \; \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H}_n[1 \; \boldsymbol{v}_n]^\mathrm{T} = 0 \wedge \mathrm{h}(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

Structured MILPs formalization

Let us define three recursive functions :

- the function f that takes a set of nodes $\mathcal{N}$ as input and returns the sum of the objectives of the nodes and their subnodes recursively,

$$f(\mathcal{N}) = \sum_{n \in \mathcal{N}} \left(\mathbf{1}^{1\times\sigma_n}\, \boldsymbol{O}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^T + f(\mathcal{N}_n) \right).$$

- the Boolean-valued function g that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{g(G)} = \overset{\text{Hyperedge}}{\boxed{[\boldsymbol{G}_{\boldsymbol{e}}[1\ \boldsymbol{v}_{\boldsymbol{e}}]^T \le 0\ \forall \mathrm{e} \in \mathcal{E}]}} \wedge \overset{\text{Node}}{\boxed{\left[\left(\boldsymbol{G}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^T \le 0 \wedge g(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right]}}.$$

- the Boolean-valued function h that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{h(G)} = [\boldsymbol{H}_{\boldsymbol{e}}[1\ \boldsymbol{v}_{\boldsymbol{e}}]^{\mathrm{T}} = 0\ \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H}_{\boldsymbol{n}}[1\ \boldsymbol{v}_{\boldsymbol{n}}]^{\mathrm{T}} = 0 \wedge \mathrm{h(G_n)} \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

Structured MILPs formalization

Let us define three recursive functions :

- the function f that takes a set of nodes $\mathcal{N}$ as input and returns the sum of the objectives of the nodes and their subnodes recursively,

$$f(\mathcal{N}) = \sum_{n \in \mathcal{N}} \left(\mathbf{1}^{1\times\boldsymbol{\sigma_n}} \boldsymbol{O_n} [1\ \boldsymbol{v_n}]^T + f(\mathcal{N}_n) \right).$$

- the Boolean-valued function g that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{g}(\mathrm{G}) = [\boldsymbol{G_e}[1\ \boldsymbol{v_e}]^T \leq 0\ \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{G_n}[1\ \boldsymbol{v_n}]^T \leq 0 \wedge g(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

- the Boolean-valued function h that takes a hypergraph $\mathrm{G} = (\mathcal{N}, \mathcal{E})$ as input and returns,

$$\mathrm{h}(\mathrm{G}) = [\boldsymbol{H_e}[1\ \boldsymbol{v_e}]^{\mathrm{T}} = 0\ \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H_n}[1\ \boldsymbol{v_n}]^{\mathrm{T}} = 0 \wedge \mathrm{h}(\mathrm{G_n}) \right) \forall \mathrm{n} \in \mathcal{N} \right].$$

Structured MILPs formalization

Given

- $f(\mathcal{N}) = \sum_{n\in\mathcal{N}} \left(\mathbf{1}^{1\times\sigma_n}\, \boldsymbol{O_n}[1\ \boldsymbol{v_n}]^T + f(\mathcal{N}_n)\right)$,
- $\mathrm{g(G)} = [\boldsymbol{G_e}[1\ \boldsymbol{v_e}]^T \le 0\ \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{G_n}[1\ \boldsymbol{v_n}]^T \le 0 \wedge g(\mathrm{G_n})\right) \forall \mathrm{n} \in \mathcal{N}\right]$,
- And, $\mathrm{h(G)} = [\boldsymbol{H_e}[1\ \boldsymbol{v_e}]^\mathrm{T} = 0\ \forall \mathrm{e} \in \mathcal{E}] \wedge \left[\left(\boldsymbol{H_n}[1\ \boldsymbol{v_n}]^\mathrm{T} = 0 \wedge \mathrm{h(G_n)}\right) \forall \mathrm{n} \in \mathcal{N}\right]$.

We have that our problems can be written as,

$$\begin{aligned} &\min \mathrm{f}\left(\mathcal{N}_g\right) \\ &\text{s.t. } h\left(G_g\right) \text{ is true} \\ &\qquad \mathrm{g}\left(G_g\right) \text{ is true} \end{aligned}$$

How can we exploit this structure in a modelling tool?

Modelling tool workflow

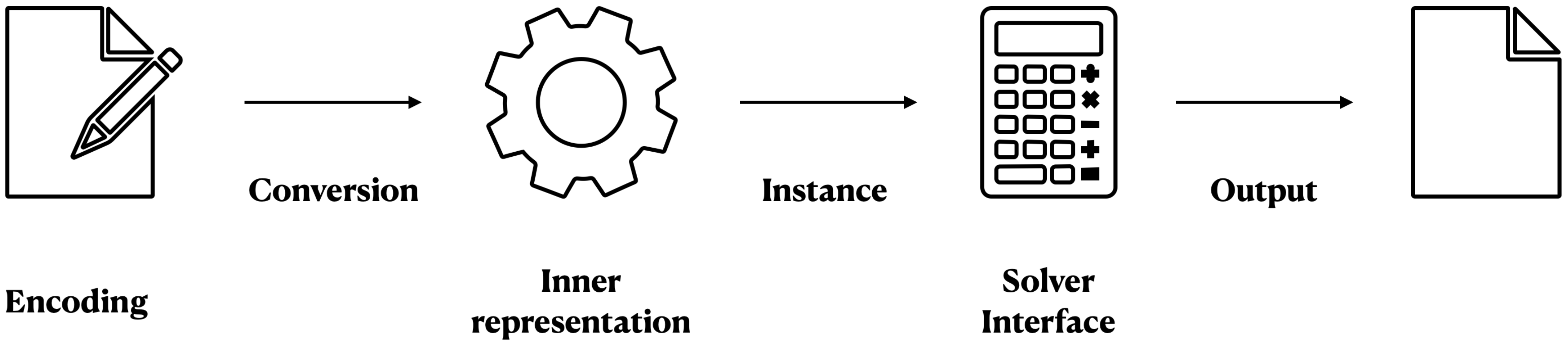


Figure 4: Modelling tool workflow

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

The Graph-Based Optimization Modelling Language (GBOML)

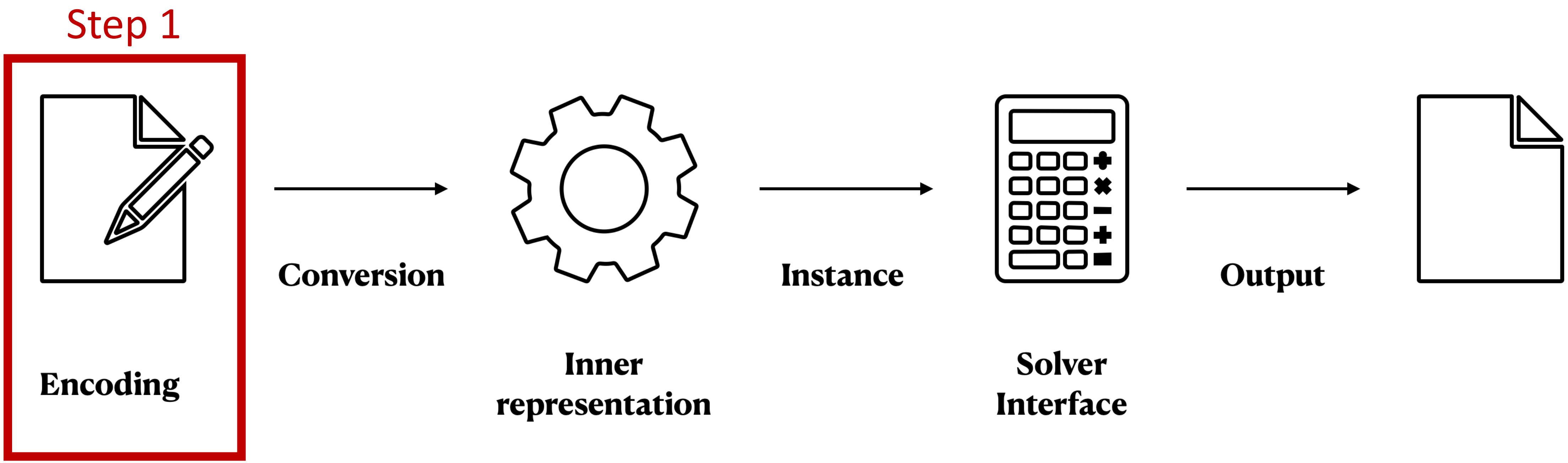


Figure 4: Modelling tool workflow

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Encoding structure [1]

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

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

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

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Encoding structure [1]

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

```
#TIMEHORIZON
T = <value>;

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

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Encoding structure [1]

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

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

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

Encoding hierarchical structure [1]

```
#NODE <node_name>
#PARAMETERS
<param_def>
[#NODE <node_name]
[#HYPEREDGE <edge_name>]
#VARIABLES
<var_def>
#CONSTRAINTS
<constr_def>
#OBJECTIVES
<obj_def>
```

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

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

The Graph-Based Optimization Modelling Language (GBOML)

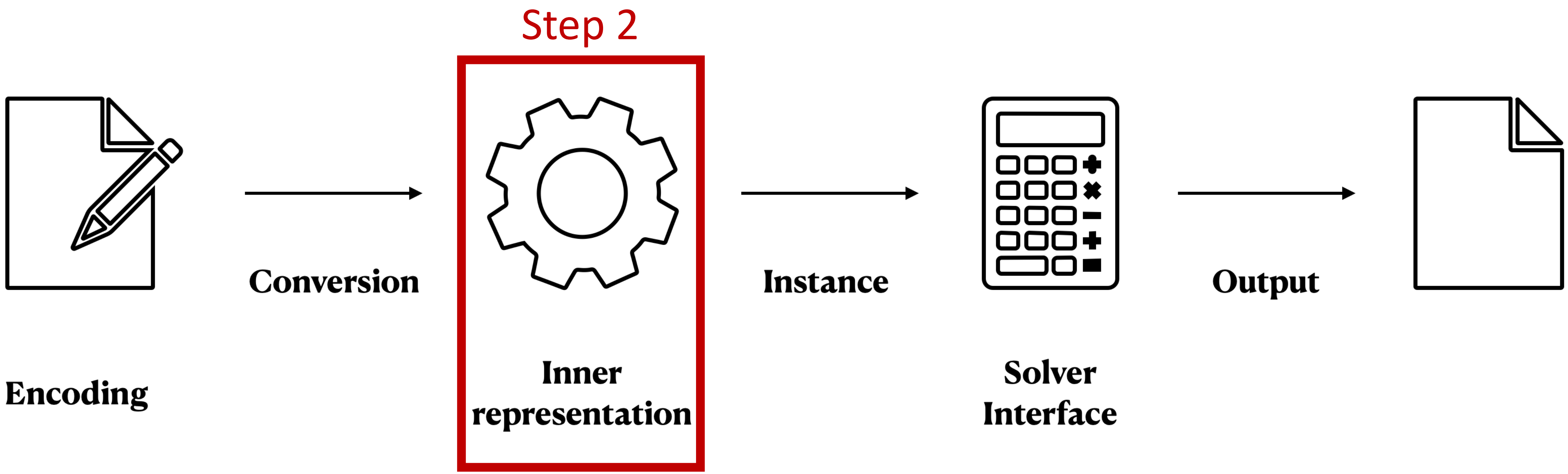


Figure 4: Modelling tool workflow

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Inner representation

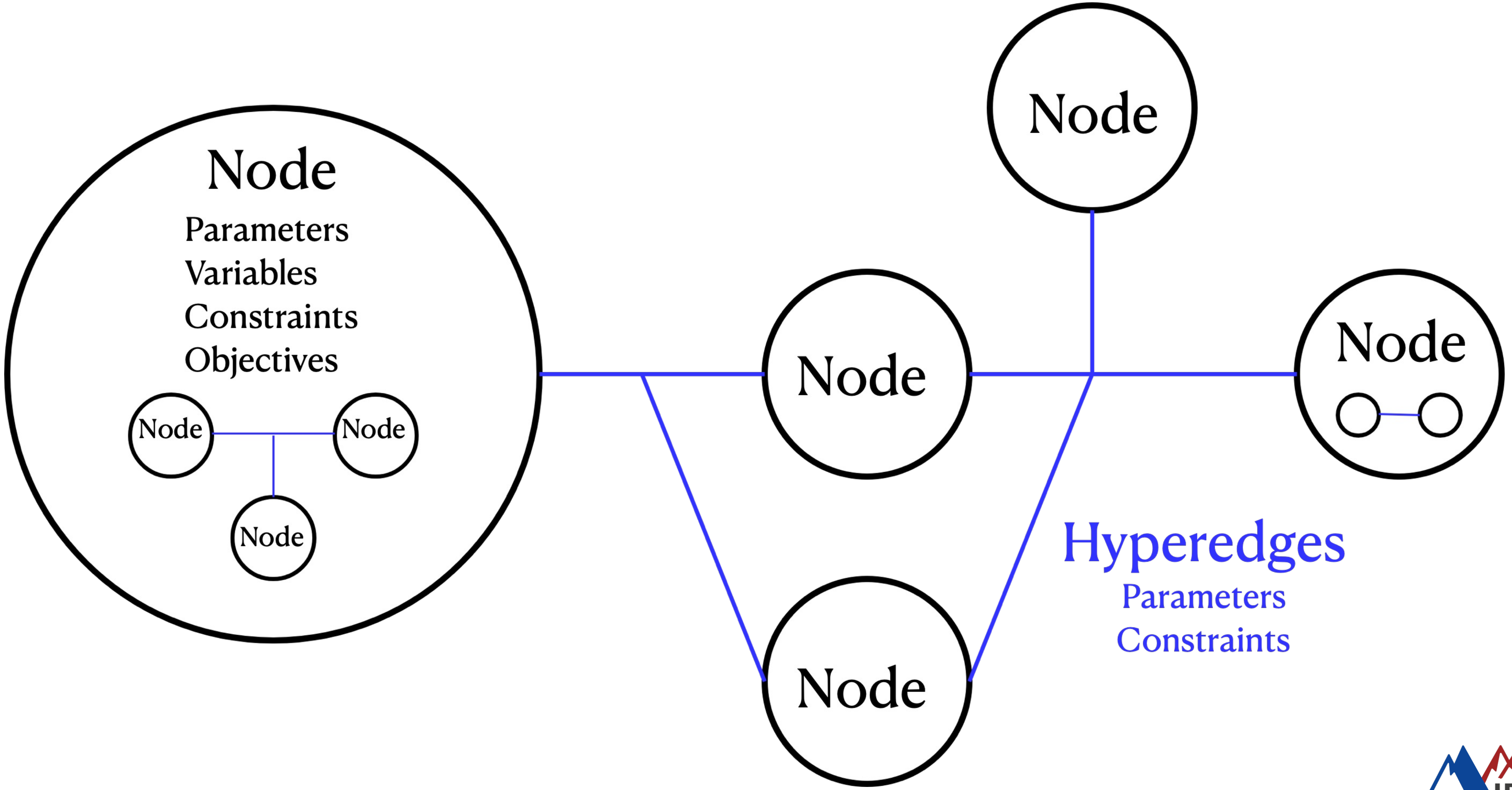


Figure 5: Hierarchical hypergraph inner representation

Inner representation

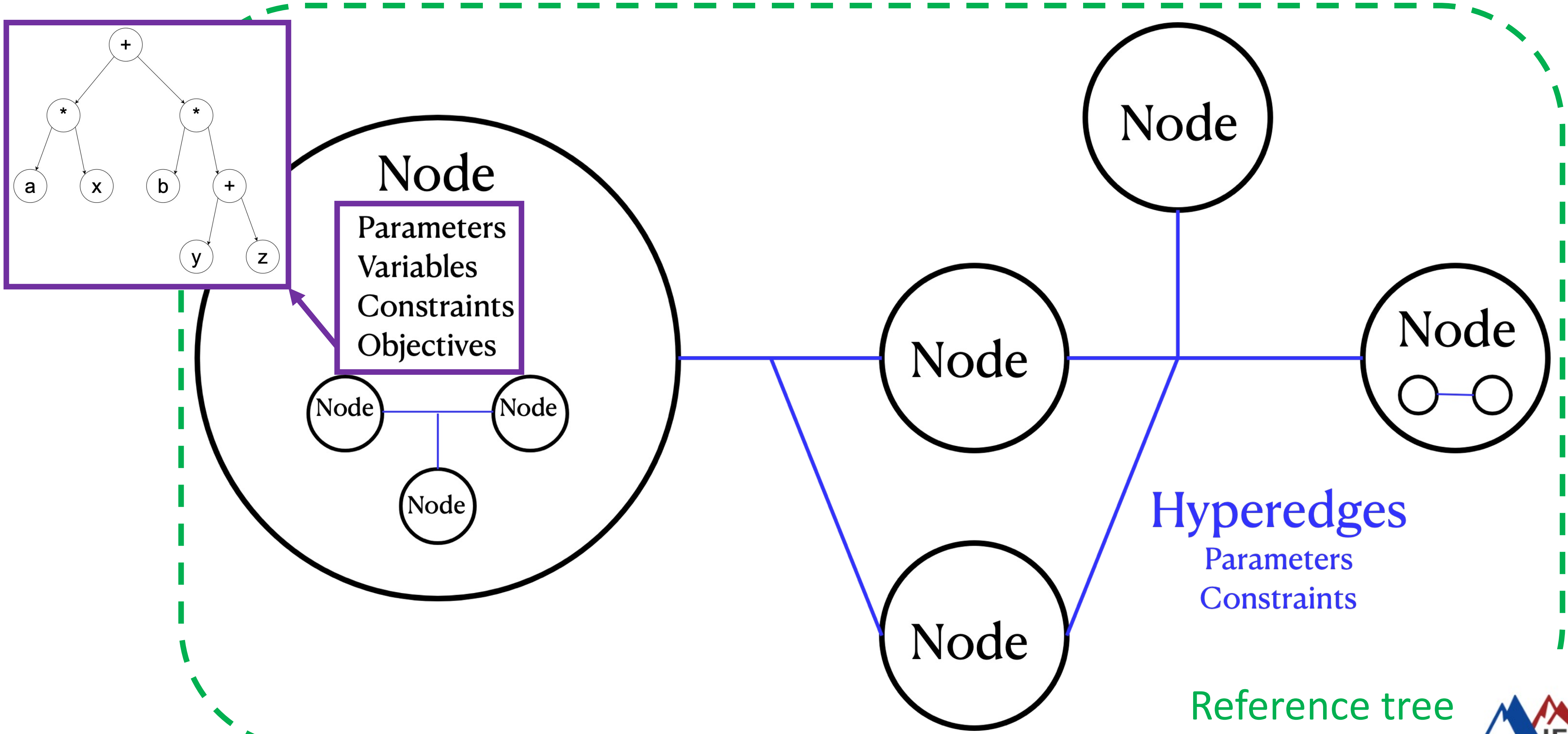


Figure 5: Hierarchical hypergraph inner representation

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Inner representation

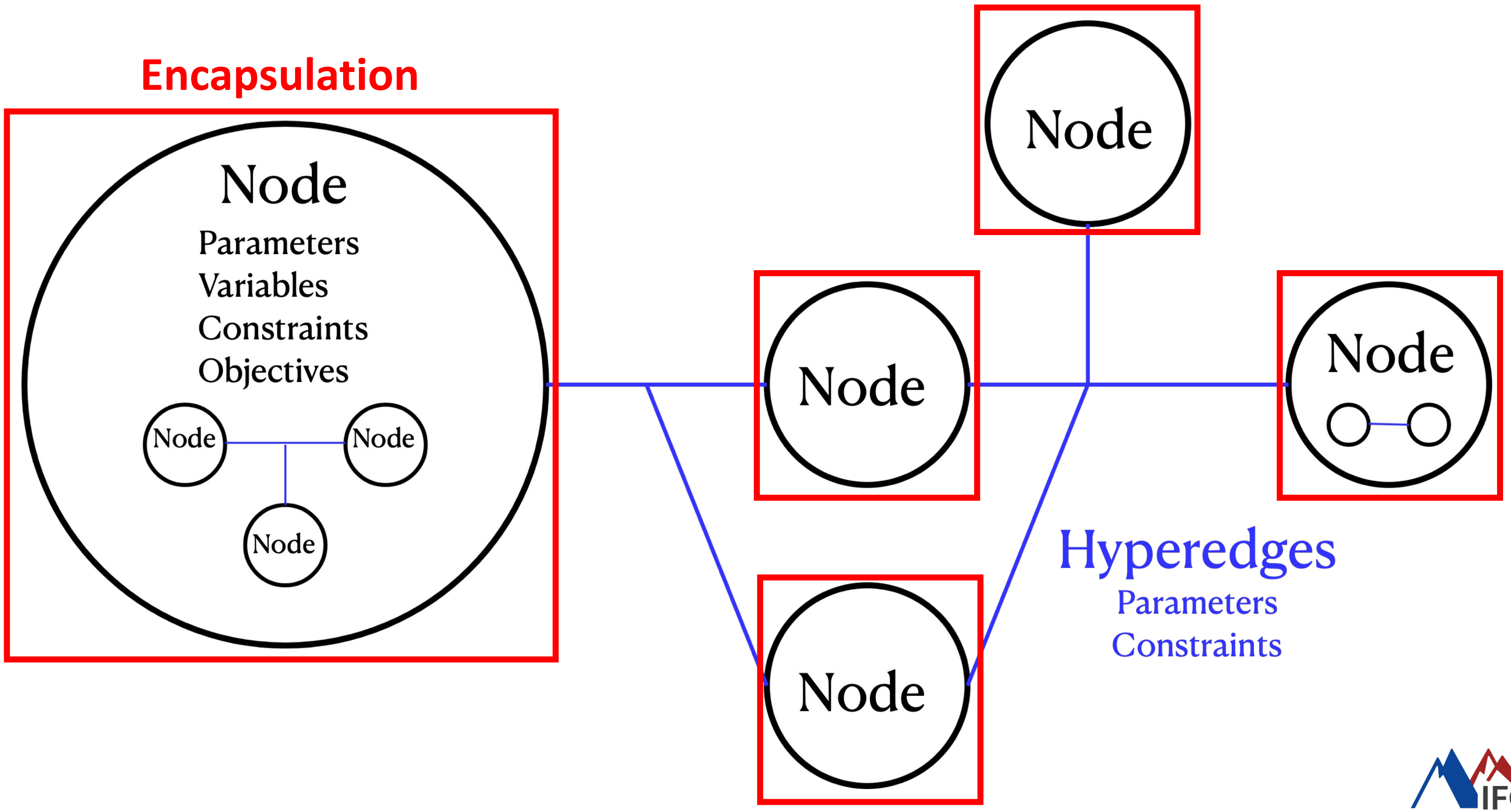


Figure 5: Hierarchical hypergraph inner representation

The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Inner representation

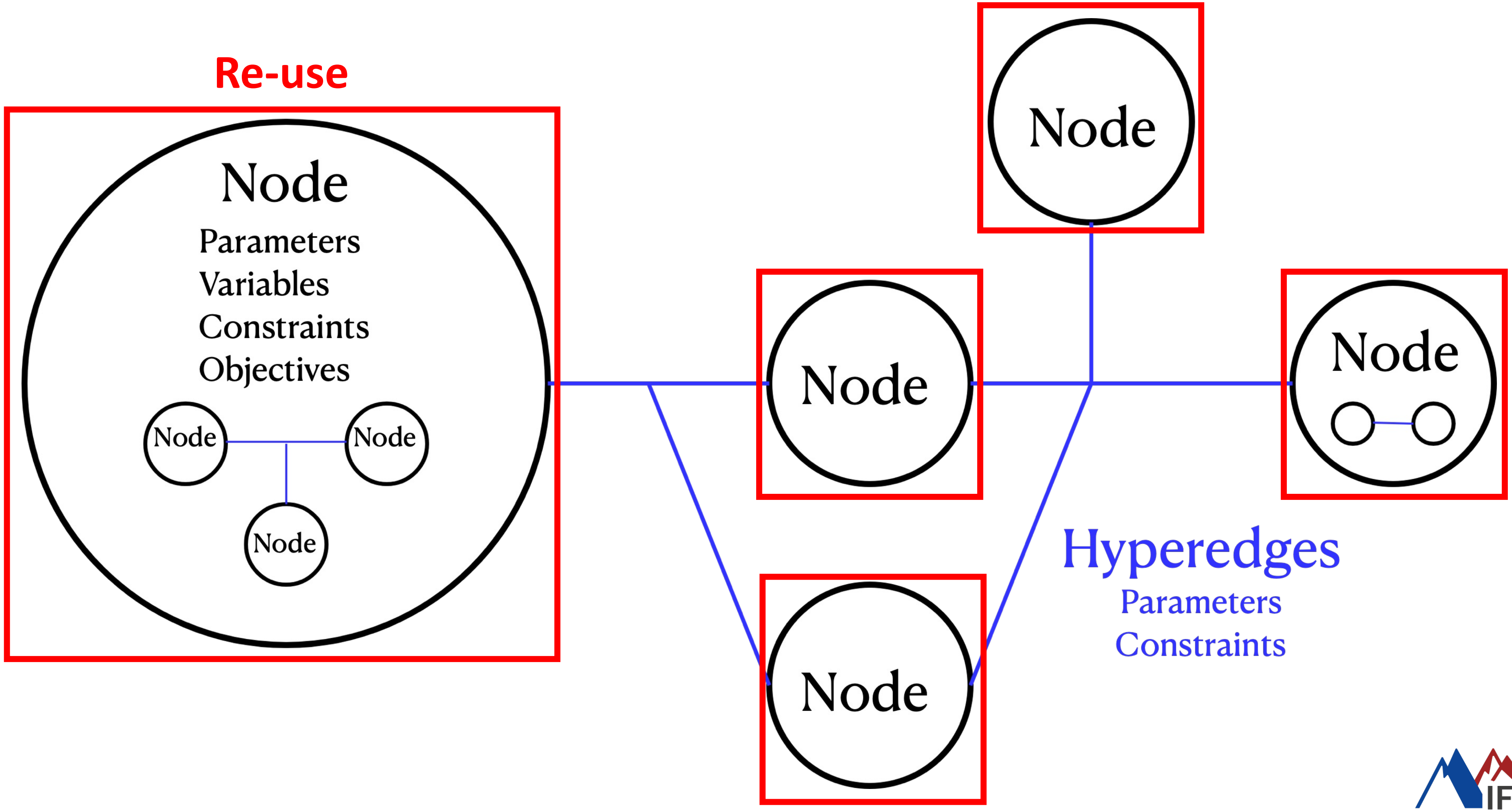


Figure 5: Hierarchical hypergraph inner representation

Inner representation

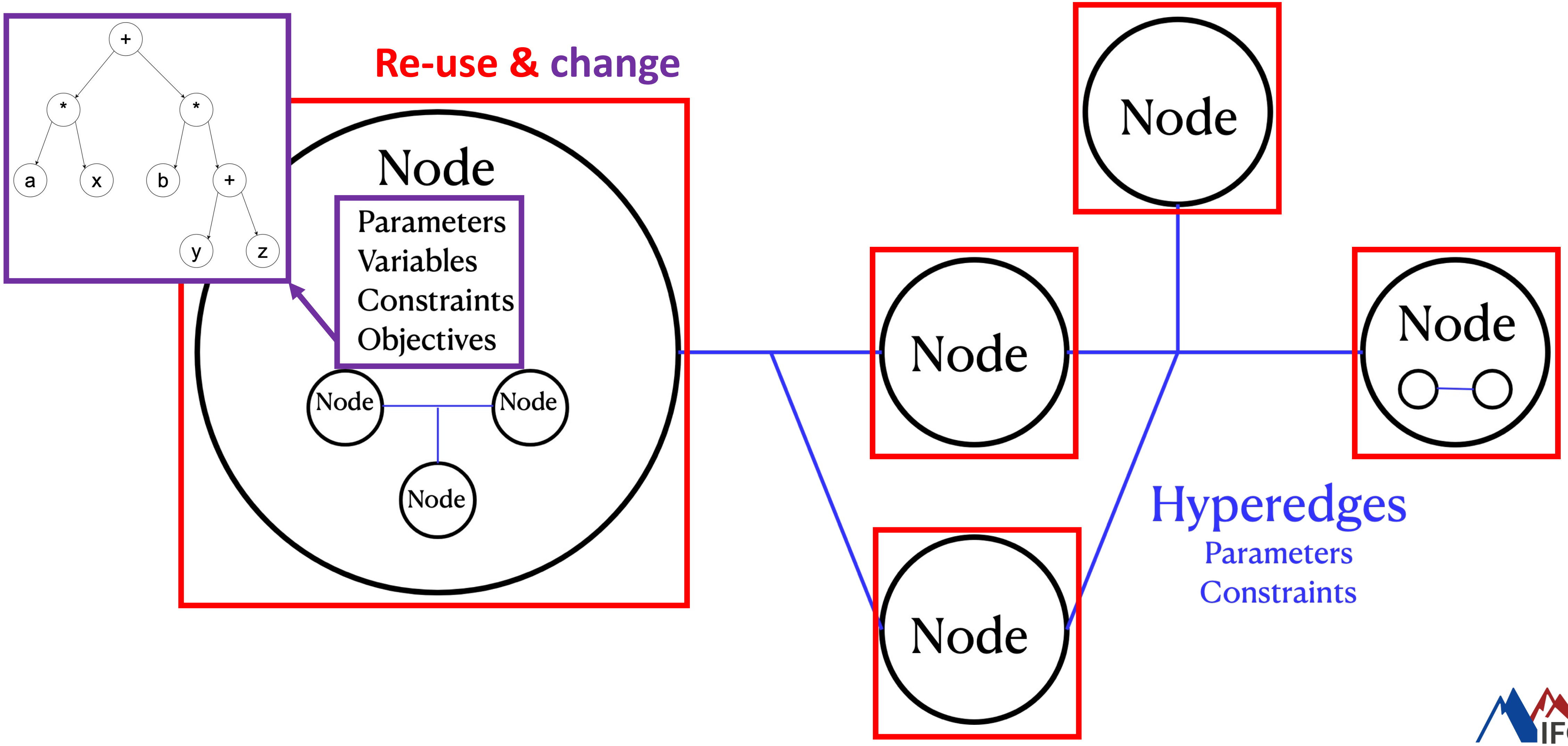


Figure 5: Hierarchical hypergraph inner representation

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Inner representation

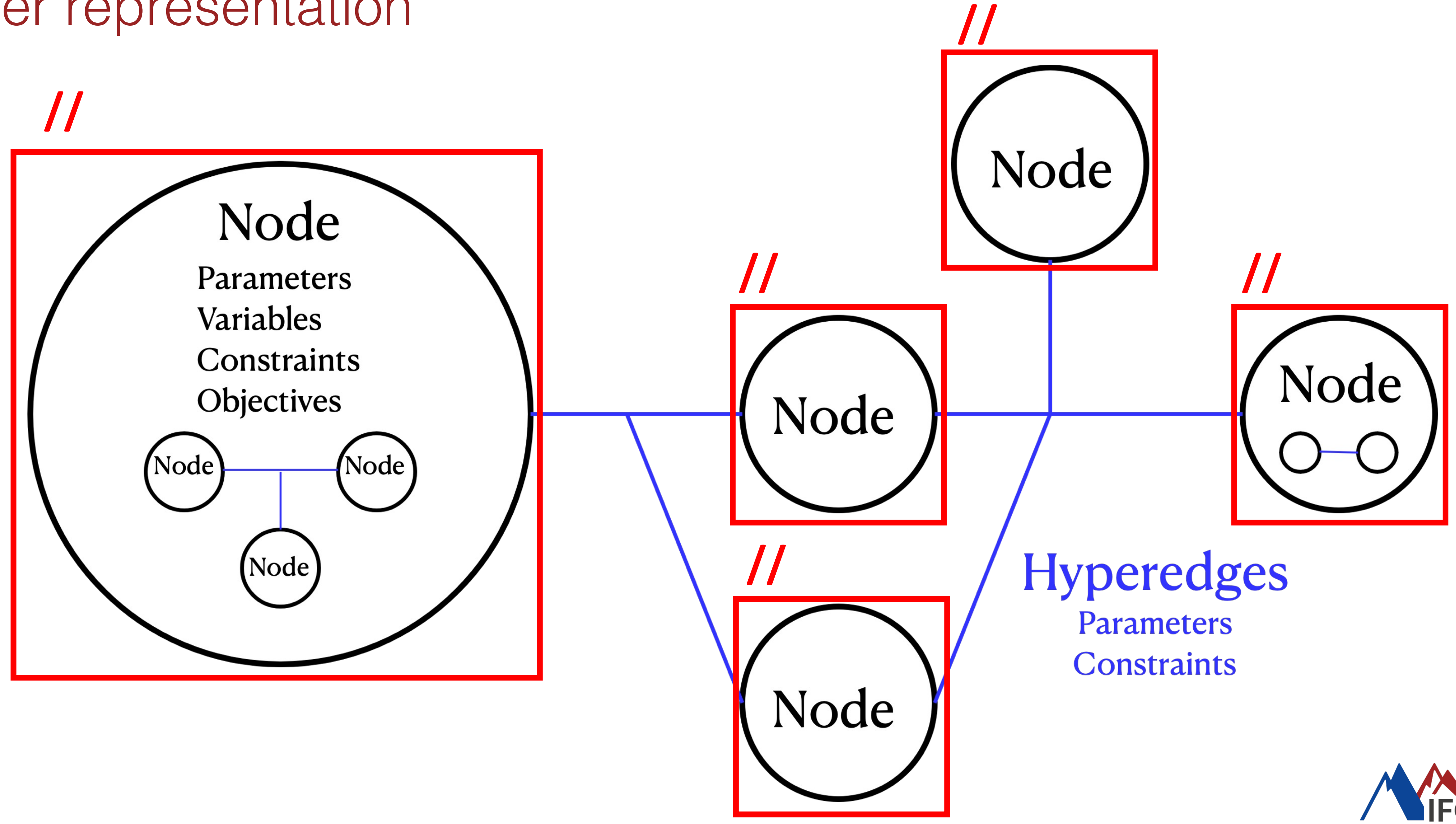


Figure 5: Hierarchical hypergraph inner representation

The Graph-Based Optimization Modelling Language (GBOML)

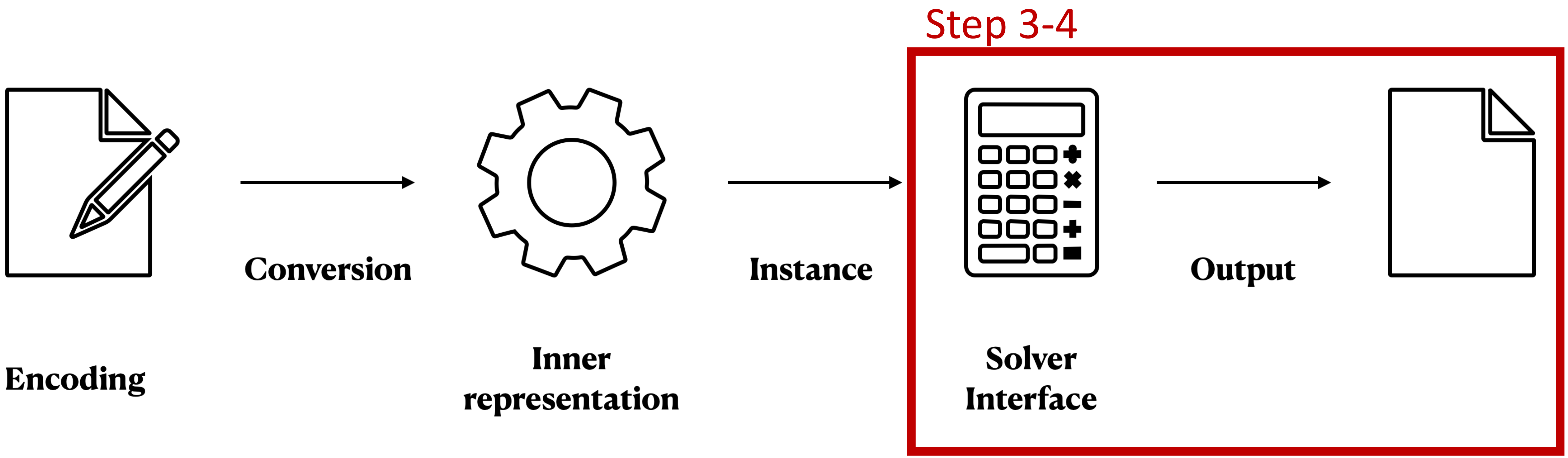


Figure 4: Modelling tool workflow

Solver interface and output

- Commercial solvers

GUROBI OPTIMIZATION [2] FICO [3] IBM [4]

- Open-source solvers

HiGHS [5] COIN|OR [6]

- Structure exploiting methods
  - DSP[7]: Dantzig-Wolfe decomposition
  - CPLEX: Benders decomposition

- Generates structure JSON and plain CSV

Benchmarking

Performances for generating an instance

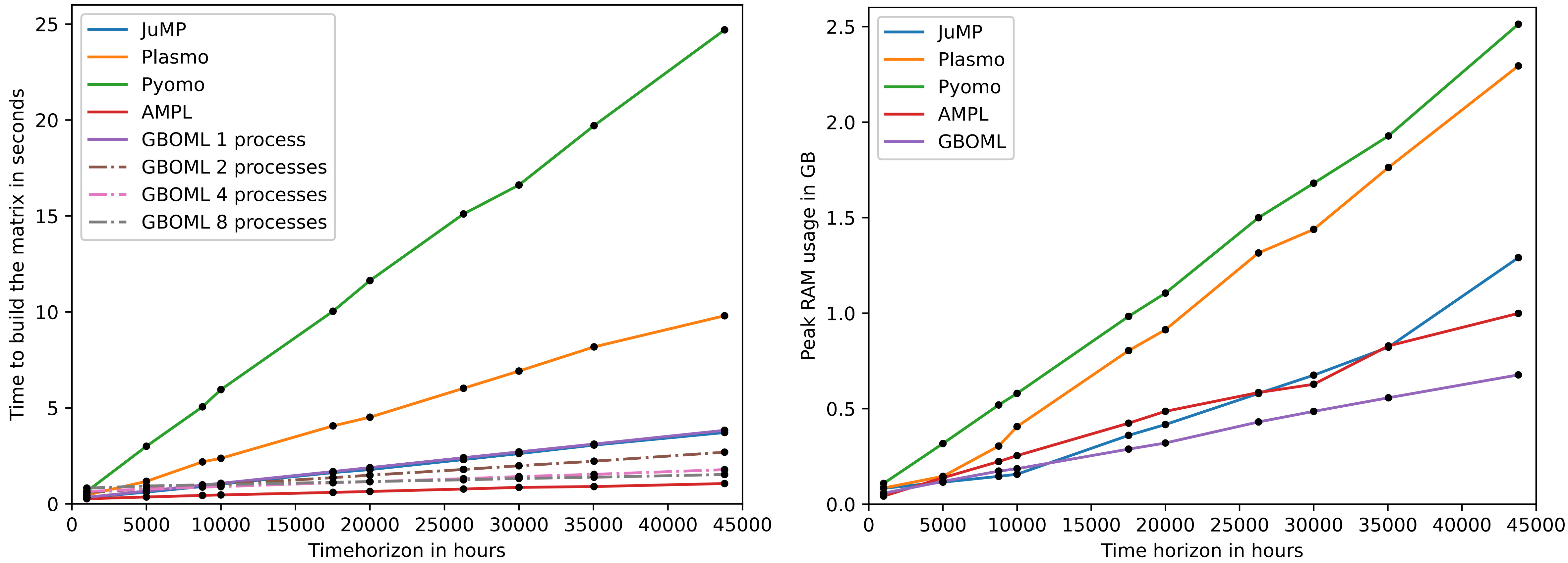


Figure 6: (Left) time taken to generate an instance (right) peak RAM usage to generate an instance – both were done on the remote renewable energy hub[8] for a growing time horizon [9]

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Structure Exploiting methods

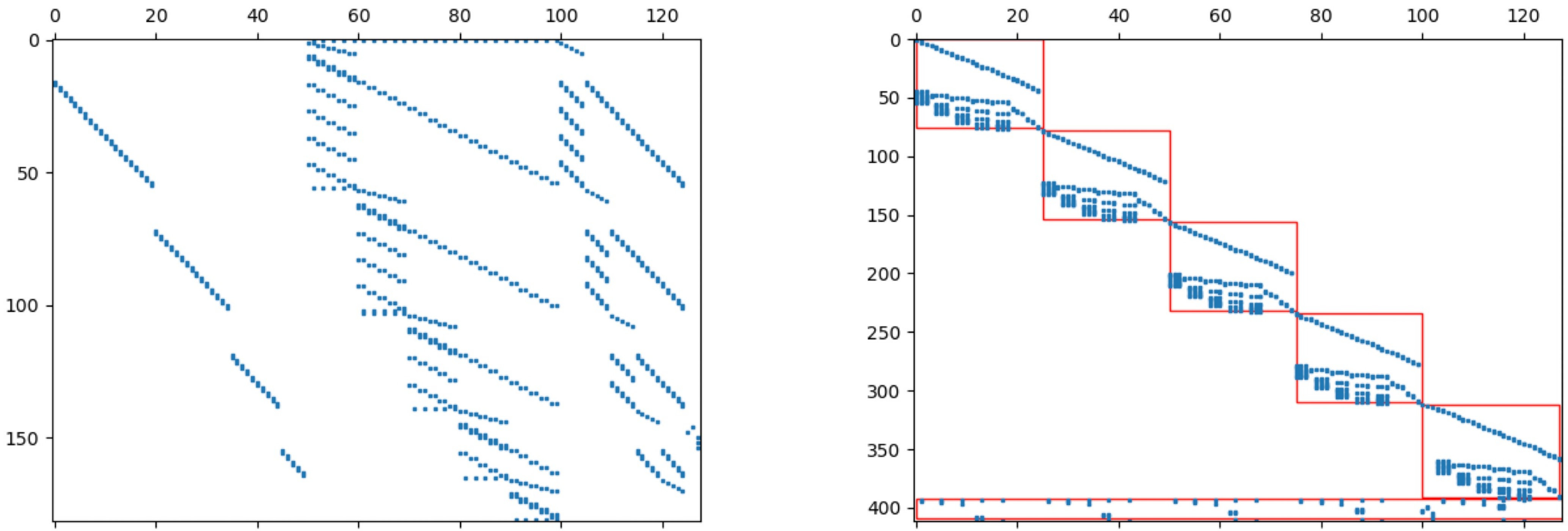


Figure 7: (Left) Plain no-swot problem problem from the MIPLIB[10] solved in 25s by Gurobi (Right) structured no-swot problem encoded in GBOML with a good decomposition solved by Dantiz-Wolfe in 2.5s.

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

Conclusion

- Structure can help
  - Encode problems more naturally
  - Enable re-use and component assembling
  - Generate models faster
  - Interface with structure exploiting methods
- In terms of tool, GBOML exploits structure from model encoding to output

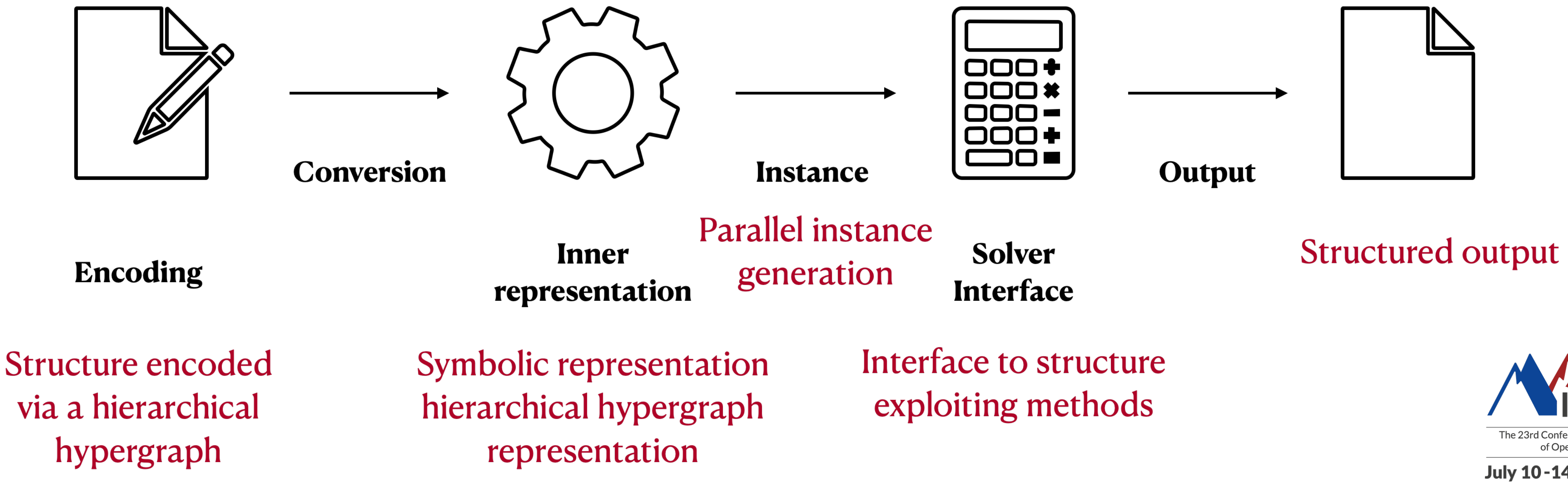


Figure 7: GBOML workflow

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

References

- [1] Bardhyl Miftari et al., ”GBOML: Graph-Based Optimization Modeling Language”, https://joss.theoj.org/papers/10.21105/joss.04158, 2022
- [2] Gurobi Optimization, LLC. All Rights Reserved. https://www.gurobi.com/
- [3] FICO® Xpress Optimization. https://www.fico.com/en/products/fico-xpress-optimization
- [4] IBM ILOG CPLEX Optimizer. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer
- [5] HiGHS - high performance software for linear optimization. https://highs.dev/
- [6] CBC/CLP from COIN-OR Foundation, Inc..https://www.coin-or.org/
- [7] DSP, Argonne National Laboratory. https://github.com/Argonne-National-Laboratory/DSP
- [8] 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
- [9] Bardhyl Miftari et al., ”GBOML: a Structure-exploiting Optimization Modeling Language in Python”, https://orbi.uliege.be/handle/2268/296930, 2022
- [10] MIPLIB, https://miplib.zib.de/

IFORS2023
The 23rd Conference of the International Federation of Operational Research Societies
July 10-14 · SANTIAGO, CHILE

ADVANCED ANALYTICS FOR A BETTER WORLD

IFORS2023

The 23rd Conference of the International Federation of Operational Research Societies

July 10-14 · SANTIAGO, CHILE

ICHIO INSTITUTO CHILENO DE INVESTIGACIÓN OPERATIVA

ALIO

UNIVERSIDAD DE CHILE

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

ISCI INSTITUTO SISTEMAS COMPLEJOS DE INGENIERÍA

Appendix : GBOML inner working