SMS++: a Structured Modelling System for Optimization

Antonio Frangioni
frangio@di.unipi.it

Rafael Durbano Lobato
rafael.lobato@di.unipi.it

Department of Computer Science, University of Pisa, Italy

PGMO Days 2019
December 4, 2019, EDF'Lab Palaiseau, France

Outline

- The plan4res project
- SMS++
- Stochastic SMS++
- Current and future work

The plan4res H2020 project

- "An end-to-end planning and operation tool, composed of a set of optimization models based on an integrated modelling of the European Energy System"

- An accurate depiction of long-term effects of strategic choices on the pan-European Energy System $\equiv$

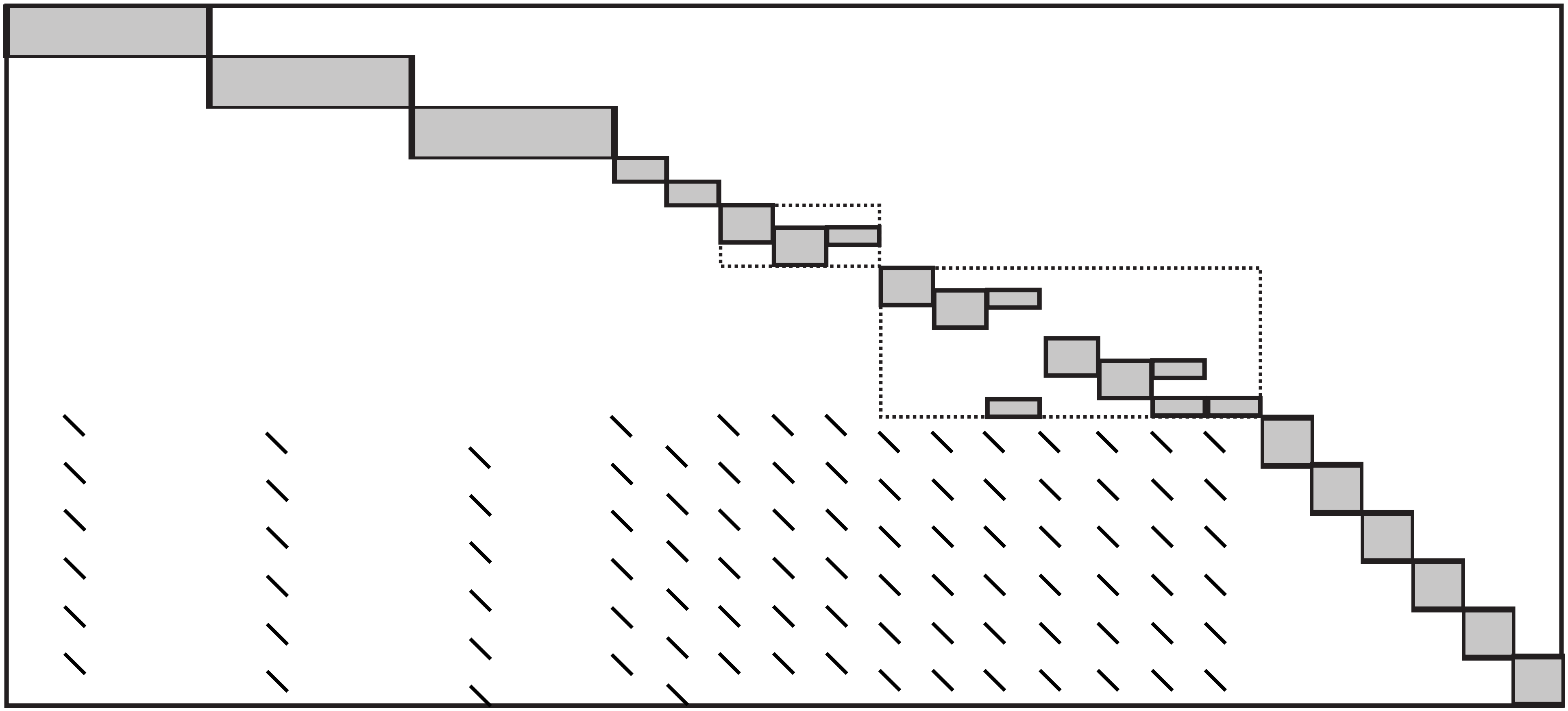
  modelling the next 30 years with 1h timescale
  and huge amounts of uncertainty over everything

- An unfeasibly large optimization problem with lots of structure

Short-term problem: Unit Commitment (UC)

- Find a (near-)optimal schedule of a large number of units satisfying the demand at each node of the network, while respecting a set of technical constraints, at each time instant of the horizon (e.g., 1 hour)
- Three natural sources of structure: unit, time, and network
- Relaxing demand constraints decomposes by unit and network: one problem per unit across all horizon and a network problem per time instant

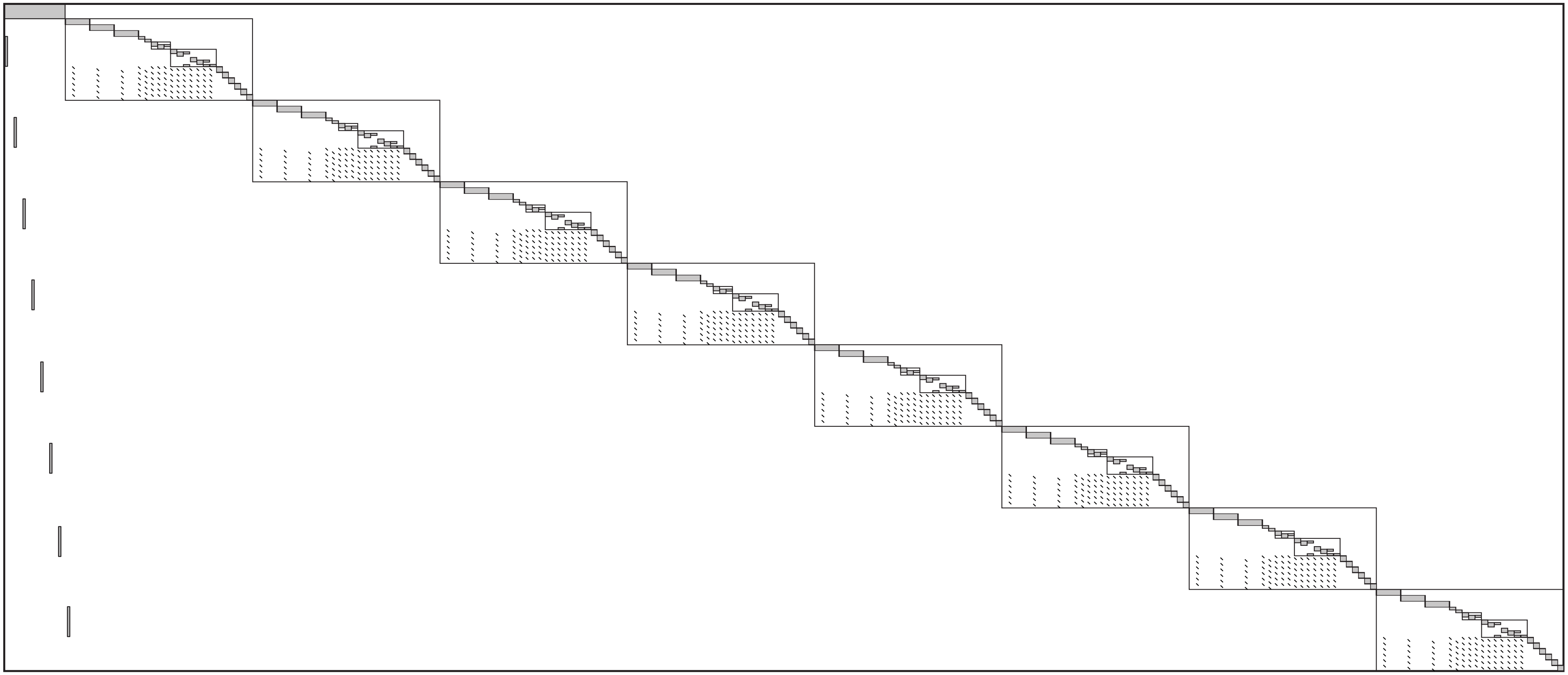
Short-term problem: Unit Commitment (UC)



Mid-term problem: Seasonal storage valuation

- UC is a (deterministic) short-term problem and lacks long-term strategies
- The mid-term (e.g., 1 year) problem provides the UC with approximations of the cost-2-go function
- UC then arises at each stage (e.g., 1 week) of the mid-term problem
- Uncertainties: inflows, demand, outages, intermittent generation
- A multi-stage stochastic optimization problem

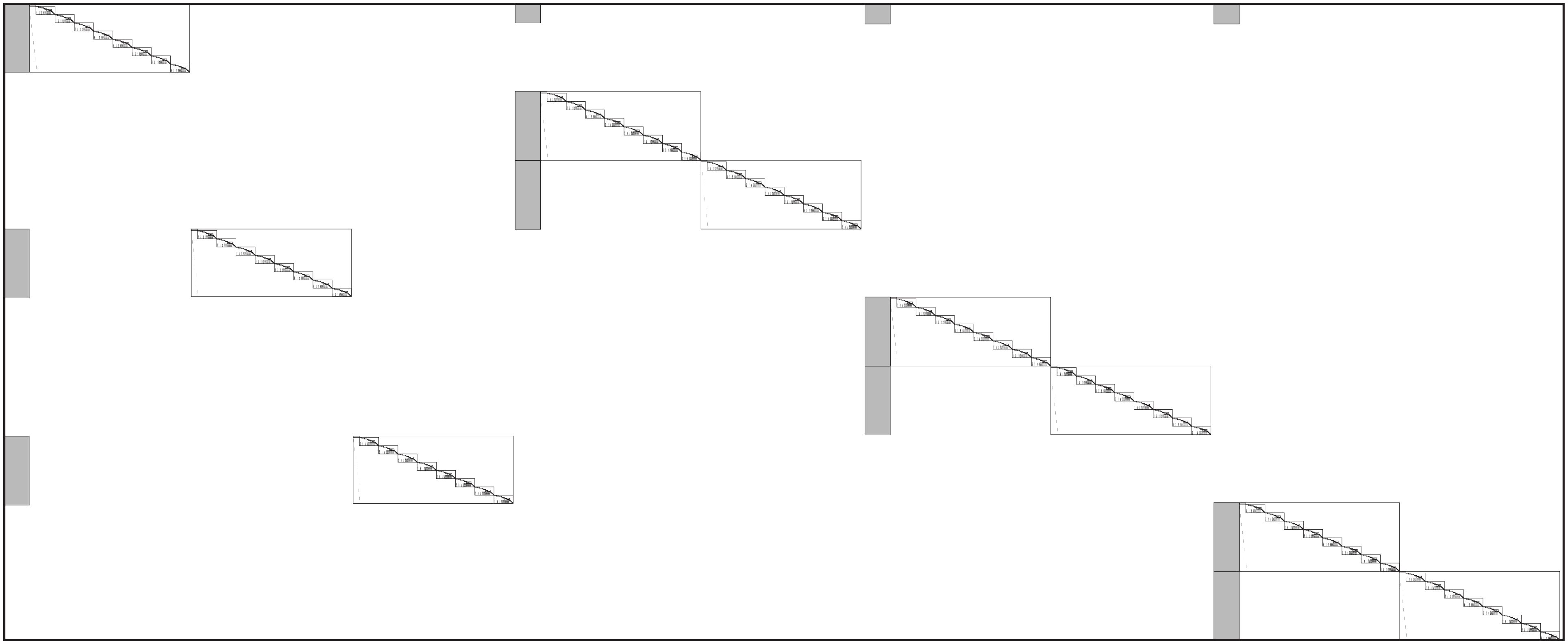
Mid-term problem: Seasonal storage valuation

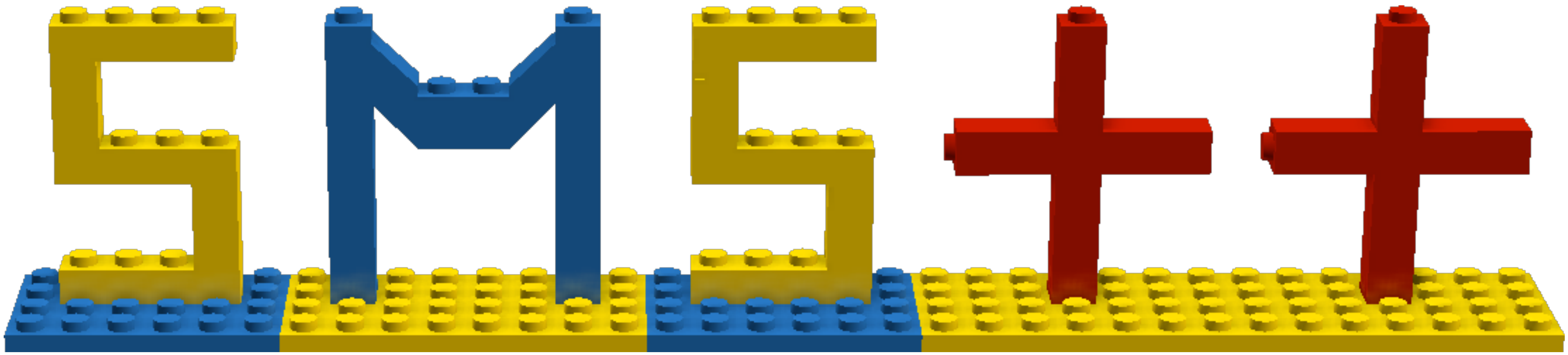


Long-term problem: Investment layer

- Long-term planning needed: the energy system changes frequently, but modifications are slow and costly
- Uncertainties in demand and production:
  - shifts in consumption patterns (EV, cryptocurrencies, . . .)
  - regulatory factors (EU energy market, . . .),
  - political factors (CO_2 emission treaties, nuclear power, . . .)
  - . . .
- Design the optimal generation mix with the optimal transmission and distribution grid capacities
- 30-year horizon with 5-year steps (multi-level recourse), many scenarios

Long-term problem



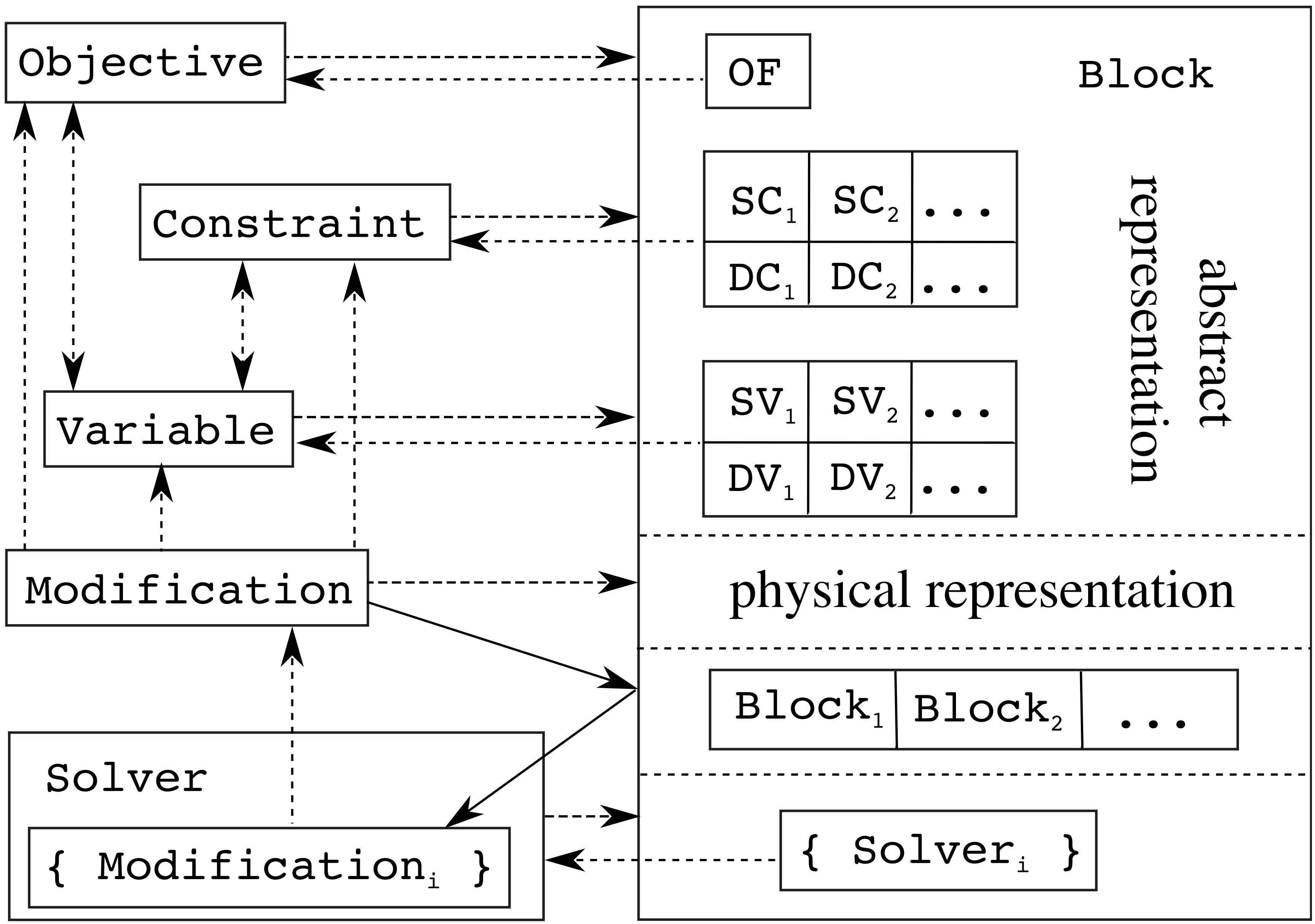

SMS++

SMS++

A set of C++ classes implementing a modelling system that:

- explicitly supports the notion of block ≡ nested structure
- separately provides "semantic" information from "syntactic" details (objective and list of constraints/variables ≡ one specific formulation among many)
- allows exploiting specialised solvers on blocks with specific structure
- manages dynamic changes in the model beyond "just" generation of constraints/variables
- manages reformulation/restriction/relaxation

SMS++


Objective
Constraint
Variable
Modification
Solver
{ Modification$_i$ }
OF
Block
SC$_1$ SC$_2$ ...
DC$_1$ DC$_2$ ...
SV$_1$ SV$_2$ ...
DV$_1$ DV$_2$ ...
abstract representation
physical representation
Block$_1$ Block$_2$ ...
{ Solver$_i$ }

SMS++
`Block`

- `Block` is an abstract class representing the general concept of "a part of a mathematical model with a well-understood identity".
- Each `:Block` is a model with specific structure (e.g., `MCFBlock:Block` = a Min-Cost Flow problem).
- Physical representation of a `Block`: whatever data structure is required to describe the instance (e.g., for a `MCFBlock`, a graph, source and sink nodes, cost and capacity of each arc, . . .)
- Abstract representation of a `Block`: an `Objective` and an "unstructured" list of `Constraints` and `Variables`.

SMS++
Block and Solver

- :Solver for a specific :Block can use the physical representation
  $\Longrightarrow$ no need for explicit Constraint or Objective
  $\Longrightarrow$ abstract representation of Block only constructed on demand

- A general-purpose Solver uses the abstract representation

- Dynamic Variable/Constraint can be generated on demand (user cuts/lazy constraints/column generation)

- Objective of sub-Blocks summed to that of father Block if it has same sense, otherwise $\min / \max$

SMS++
Solver

- Solver = interface between a Block and algorithms solving it
- Each Solver attached to a single Block, from which it picks all the data, but any # of Solver can be attached to the same Block
- Individual Solver can be attached to sub-Block of a Block
- Tries to cater for all the important needs:
  - optimal and sub-optimal solutions, provably unbounded/infeasible
  - time/resource limits for solutions, but restarts (reoptimization)
  - any # of multiple solutions produced on demand
  - lazily reacts to changes in the data of the Block via Modification

- Most Block components can change, but not all:
  - set of sub-Block
  - # and shape of groups of Variable/Constraint
- Any change is communicated to each interested Solver (attached to the Block or any of its ancestor) via a Modification object
- However, two different kinds of Modification (what changes):
  - physical Modification, only specialized Solver concerned
  - abstract Modification, only Solver using it concerned

Example: Capacitated Facility Location

Given a set L of locations and a set D of customers, the problem consists in selecting a subset of the locations in which facilities will be placed in order to serve the given set of customers.

Example: Capacitated Facility Location

$$
\begin{aligned}
\min \quad & \sum_{i \in L} f_i y_i + \sum_{i \in L} \sum_{j \in D} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in L} x_{ij} = 1, \forall j \in D \\
& \sum_{j \in D} d_j x_{ij} \leq u_i y_i, \forall i \in L \\
& x \geq 0 \\
& y \in \{0, 1\}^{|L|}
\end{aligned}
$$

Example: Capacitated Facility Location
Abstract representation

$$
\begin{aligned}
\min \quad & f(x, y) \\
\text{s.t.} \quad & h_i(x) = 0, \forall i \in I \\
& g_j(x) \leq 0, \forall j \in J \\
& x \in X \\
& y \in Y
\end{aligned}
$$

Example: Capacitated Facility Location
Physical representation

- L and D
- $f_i,\ \forall i \in L$
- $c_{ij},\ \forall\ i \in L,\ \forall j \in D$
- $d_j,\ \forall j \in D$

Example: Capacitated Facility Location

$$
\begin{aligned}
\min \quad & \sum_{i \in L} f_i y_i \\
\text{s.t.} \quad & \sum_{j \in D} d_j x_{ij} \leq u_i y_i, \forall i \in L \\
& y \in \{0,1\}^{|L|}
\end{aligned}
$$

j-th sub-Block:

$$
\begin{aligned}
\min \quad & \sum_{i \in L} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in L} x_{ij} = 1 \\
& x_j \geq 0
\end{aligned}
$$

Stochastic SMS++

SMS++ is (almost) ready for deterministic optimization.

Stochastic SMS++

SMS++ is (almost) ready for deterministic optimization.

What about stochastic optimization?

Stochastic SMS++

We must represent uncertainty in SMS++.

Stochastic SMS++

We must represent uncertainty in SMS++.

`UCBlock`	`ThermalUnitBlock`
`UnitBlock`	`EMobilityUnitBlock`
`HeatBlock`	`PowerToGasUnitBlock`
`NetworkBlock`	`BatteryUnitBlock`
`HydroUnitBlock`	`IntermittentUnitBlock`
`DCNetworkBlock`	`CentralizedDemandResponseUnitBlock`
`BusNetworkBlock`	...

Ideally, without changing the implementation of the `Blocks`.

Stochastic SMS++

$$\min_{\text{s.t. } x_1 \in X_1} f_1(x_1) + \mathbb{E}\Bigg[\min_{\text{s.t. } x_2 \in X_2(x_1,\xi_2)} f_2(x_2;\xi_2) + \mathbb{E}_{|\xi_{[2]}}\Bigg[\cdots + \mathbb{E}_{|\xi_{[T-1]}}\Bigg[\min_{\text{s.t. } x_T \in X_T(x_{T-1},\xi_T)} f_T(x_T;\xi_T) \Bigg]\Bigg]\Bigg]$$

$\{\xi_t\}_{t \in \{2,\dots,T\}}$ is a stochastic process

Stochastic SMS++

$$\begin{array}{ll}\min & f_1(x_1)\\ \text{s.t.} & x_1 \in X_1\end{array} + \mathbb{E}\left[\begin{array}{ll}\min & f_2(x_2;\xi_2)\\ \text{s.t.} & x_2 \in X_2(x_1,\xi_2)\end{array} + \mathbb{E}_{|\xi_{[2]}}\left[\cdots + \mathbb{E}_{|\xi_{[T-1]}}\left[\begin{array}{ll}\min & f_T(x_T;\xi_T)\\ \text{s.t.} & x_T \in X_T(x_{T-1},\xi_T)\end{array}\right]\right]\right]$$

$\{\xi_t\}_{t\in\{2,\ldots,T\}}$ is a stochastic process

Stochastic dual dynamic programming (SDDP)

Stochastic SMS++

$$
\begin{aligned}
V_t(x_{t-1}, \xi_t) = \min_{x_t} \quad & f_t(x_t; \xi_t) + \mathcal{V}_{t+1}(x_t, \xi_t) \\
\text{s.t.} \quad & x_t \in X_t(x_{t-1}, \xi_t)
\end{aligned}
$$

$$
\mathcal{V}_{t+1}(x_t, \xi_t) = \mathbb{E}\left[V_{t+1}(x_t, \xi_{t+1}) \mid \xi_{[t]}\right]
$$

Stochastic SMS++

$$
\begin{aligned}
V_t(x_{t-1}, \xi_t) = \min_{x_t} \quad & f_t(x_t; \xi_t) + \mathcal{V}_{t+1}(x_t, \xi_t) \\
\text{s.t.} \quad & x_t \in X_t(x_{t-1}, \xi_t)
\end{aligned}
$$

$$
\mathcal{V}_{t+1}(x_t, \xi_t) = \mathbb{E}\left[V_{t+1}(x_t, \xi_{t+1}) \mid \xi_{[t]} \right]
$$

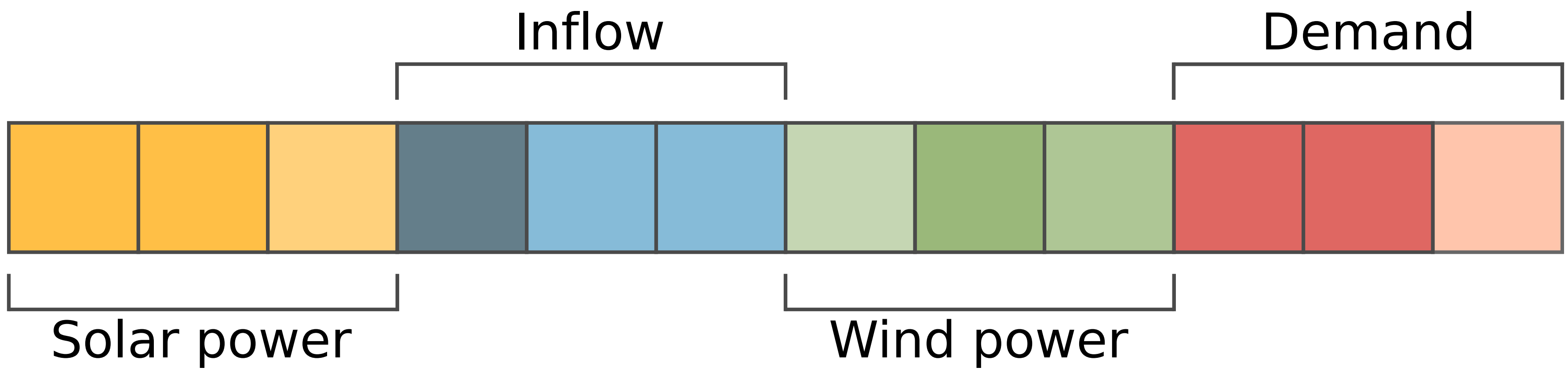
$$
\begin{aligned}
\min_{x_t} \quad & f_t(x_t; \tilde{\xi}_t) + \mathcal{P}_{t+1}(x_t) \\
\text{s.t.} \quad & x_t \in X_t(\tilde{x}_{t-1}, \tilde{\xi}_t)
\end{aligned}
$$

Stochastic SMS++

We must be able to:

- Simulate the random variables.
- Update the data of the `Blocks` for a given realization of the random variables.

Stochastic SMS++



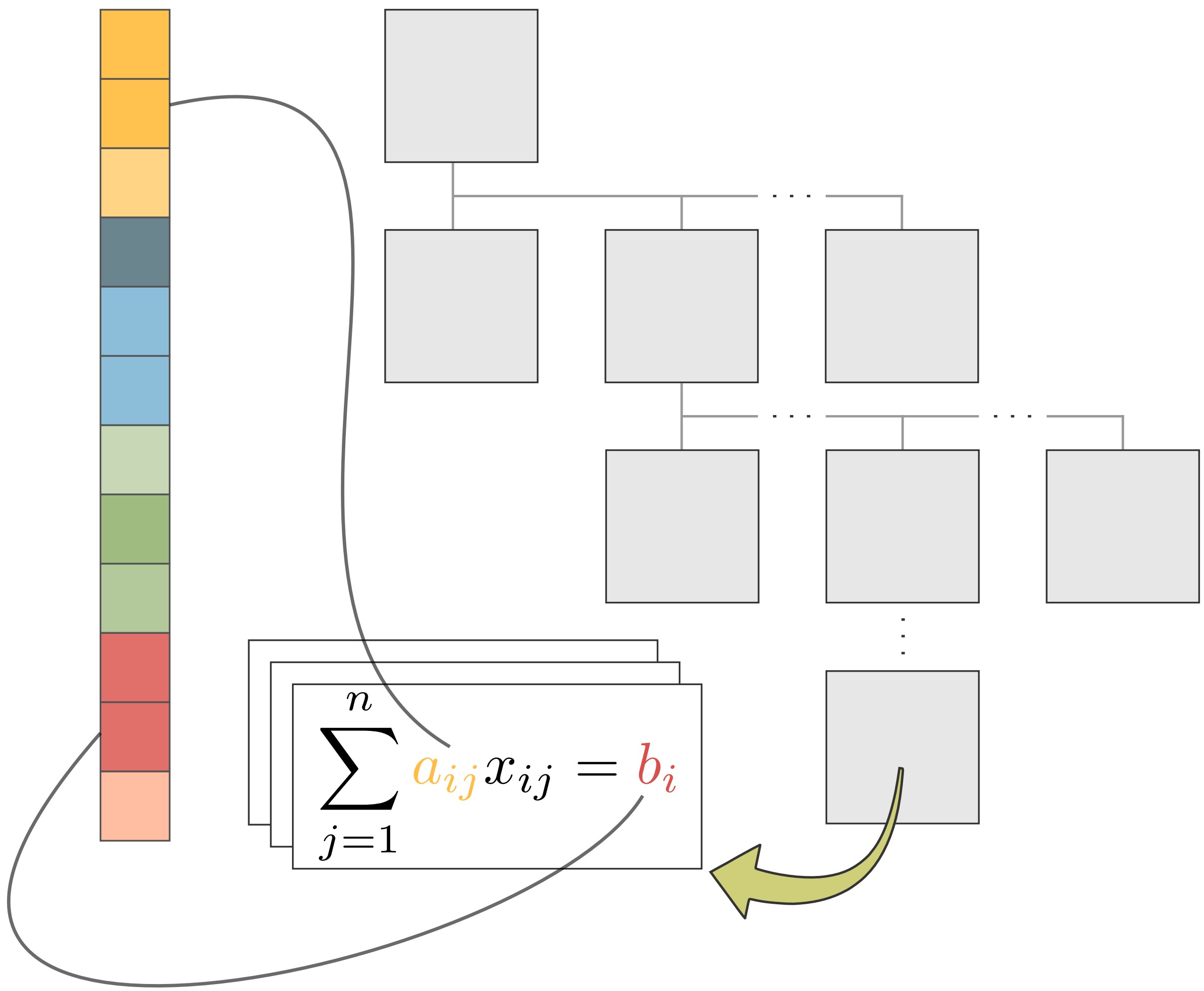
Setting the data

We can set the data by using either the abstract or the physical representation of the `Block`.

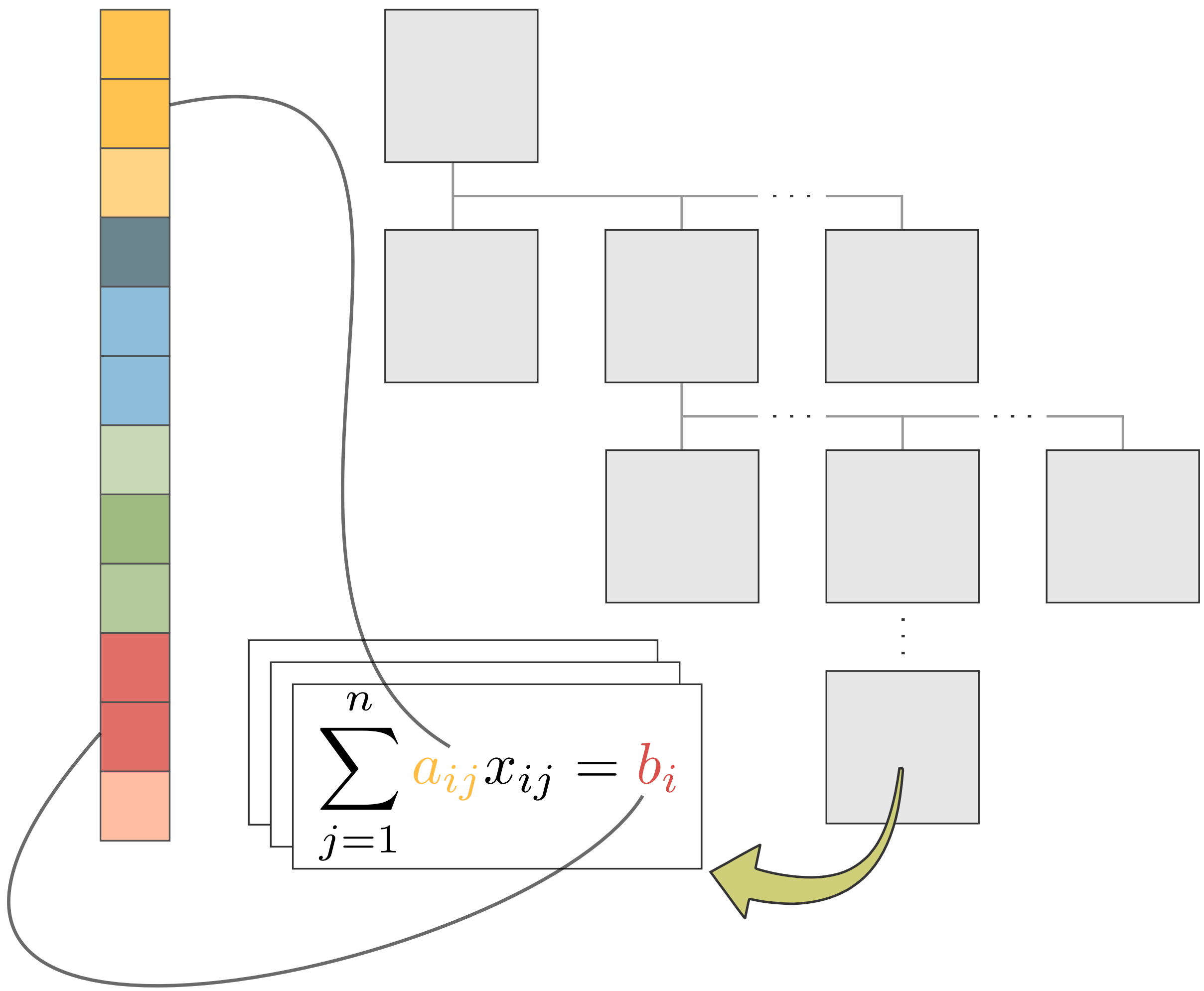
Setting the data
Abstract representation

- A random variable may appear in the `Objective` and/or in the `Constraints`.
- Let's assume that a random variable appears in the right-(or left-)hand side of a `RowConstraint` or as the coefficient of some `Variable`.

Setting the data
Abstract representation



Setting the data
Abstract representation



It is doable, but can be complicated for the user.

Setting the data
Physical representation

- We set the data by using the available methods in the `Blocks`.

```
void HydroUnitBlock::set_inflow(std::vector<double> inflow);
```

- The methods are first registered in the methods factory.
- Pointer to a method can be retrieved by its name.

Setting the data
Physical representation

- What if the available methods are not enough?
- One can write their own method and register it in the methods factory.

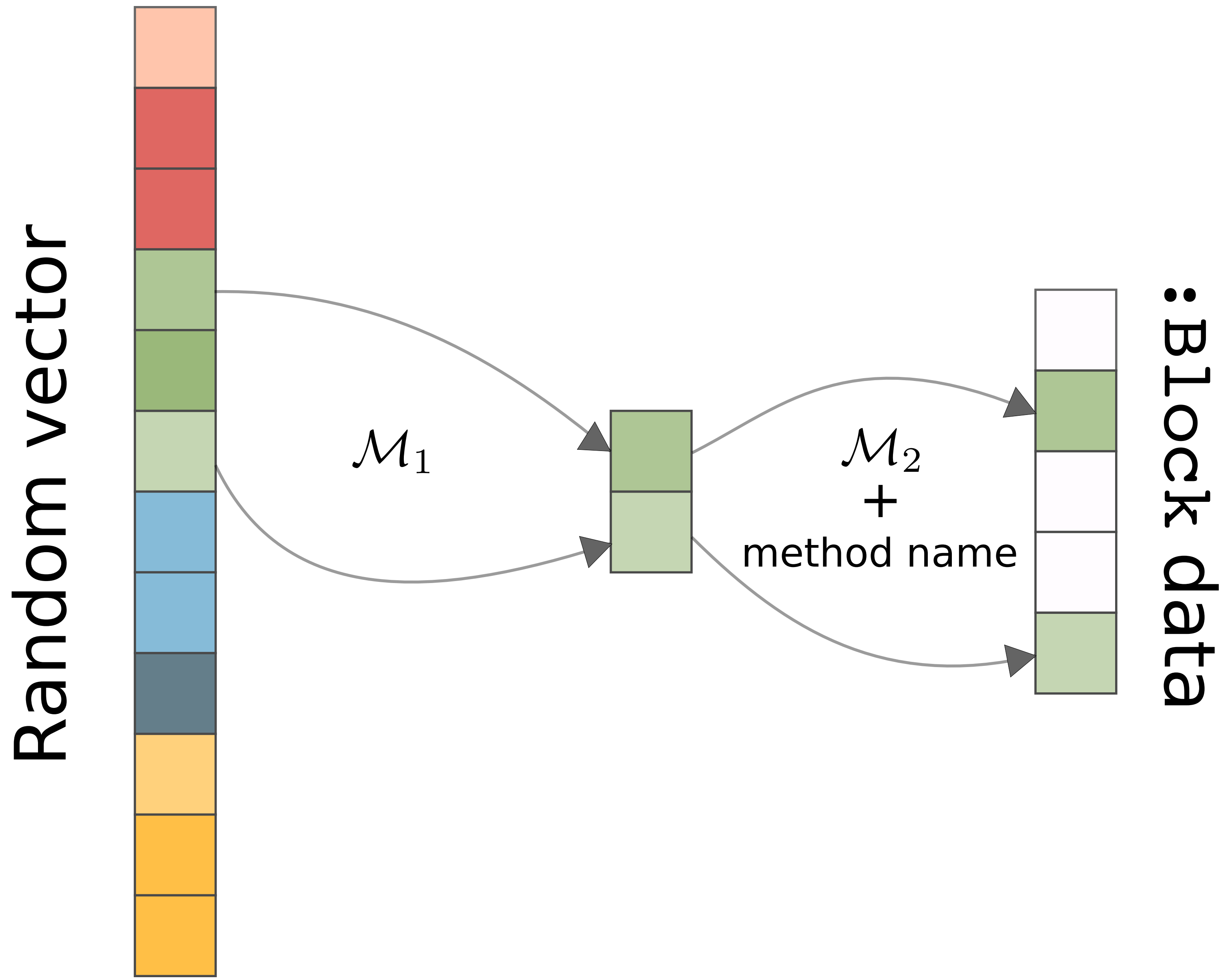
```
void customized_set_inflow(Block * block, ...) {
    ...
}

Block::register_method
  ("HydroUnitBlock::customized_set_inflow",
   new Block::FunctionType<...>(customized_set_inflow));
```

Data Mapping

- Data mapping identifies the random variables in the `Blocks`.
- At the same time, it provides means to set the values of those variables.
- It associates methods in the methods factory with instances of `:Block`.

Data Mapping



StochasticBlock

- It has a (single) nested `Block` (which is becoming stochastic).

- It has a data mapping.

- It has a probability distribution or a "partial stochastic process".

Current and future work

- `BendersBlock` turns a Block into its Benders' reformulation

- `LagrangianDualBlock` turns any `Block` into its Lagrangian Dual w.r.t. constraints linking its sub-`Block`

- Asynchronous execution of the computationally heavy parts.

Acknowledgements

Copyright © Università di Pisa 2019, all rights reserved.

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the PLAN4RES Consortium. In addition, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

This document may change without notice.

The content of this document only reflects the author's views. The European Commission / Innovation and Networks Executive Agency is not responsible for any use that may be made of the information it contains.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773897