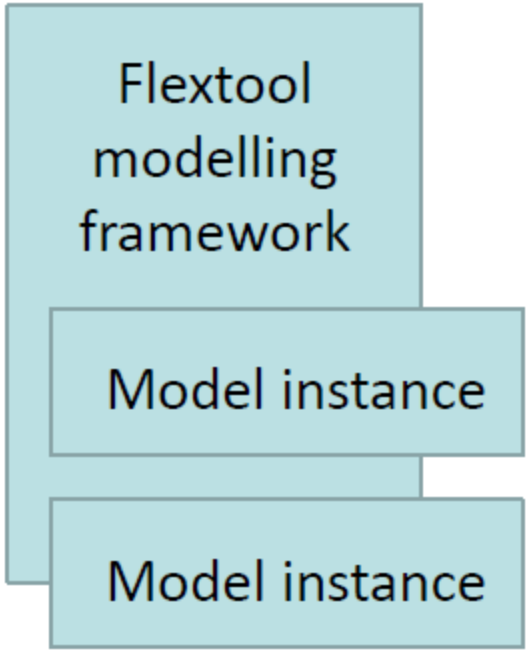
FLEXTOOL IN PRACTICE

LEAP-RE

OASES

Some basic terminology



Difference between

- ‘Modelling framework’ FlexTool. Does not contain data.
- ’Model’ or ’Model instance’, e.g. FlexTool Egypt local.

FlexTool framework for building models can be accessed via two different interfaces:

- Spine Toolbox (in this project). More versatile.
- Flextoolweb interface. Needs a server, will be simpler and more visual.

Remember, FlexTool is still a research level tool in progress so patience is needed.

Input data

Group	Main data types, e.g.
Electricity transmission data (each node)	Transmission and interconnection capacities
Generation and storage capacity data (each node)	Installed capacity, technical data of units, hydro reservoir capacity
Time-series data (e.g. 8760 hourly values, each node)	Electricity demand, hydro inflow, wind and solar profile
Cost data (annual)	Fuel prices, invest costs and emission rate
Annual system data (annual)	Capacity margin, emission limit

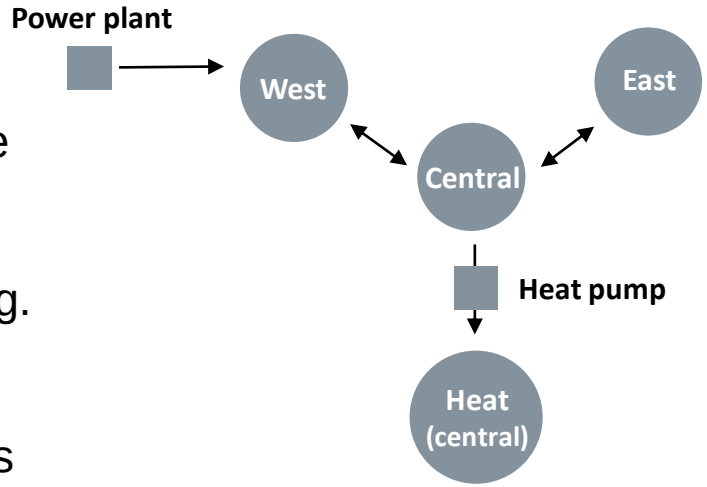
Model output

Output group	Main data types
Flexibility indicators	Loss of load, reserve shortages, spillage, VRE curtailment
Unit dispatch	Hourly dispatch of every unit
Transmission between nodes	Hourly use of transmission lines
Costs	OPEX: fuel costs, other O&M costs, CO_2 costs, penalty costs from inflexibilities CAPEX: Generation capacity investments, transmission line investments, storage investments
Marginal prices	Hourly marginal price of electricity
Investments	Invested amount, type, and costs for generation, transmission, and storages

Node-unit and alternative-scenario

FlexTool has three basic building blocks:

- Nodes (n): Each location and energy vector is described as one or more nodes, e.g. electricity grid divided to three nodes (west, central, and east).
- Unit (u):Units can produce, consume and store from a node in a grid, e.g. power plant produces electricity to the west node of the electricity grid. Units can also be used to convert energy between grids.
- Connections (u-n / n-n): Connections can exist between units and nodes or between nodes. E.g. electricity transimissionlines between nodes.

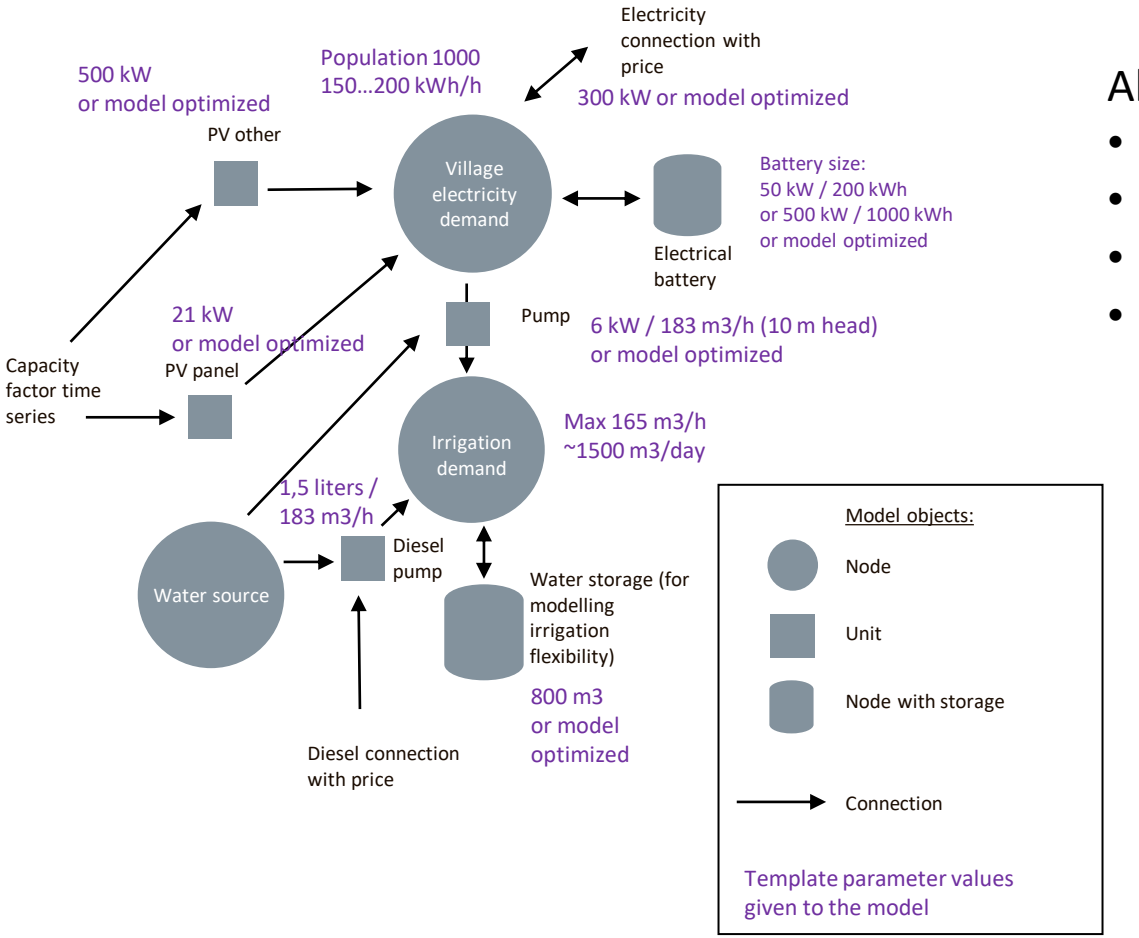


Alternatives are used to give alternative values for the same data items

- Scenarios are built from the alternatives
- To make a scenario: Layer any number of alternatives on top of each other

Example of node-unit and alternative-scenario

Egypt local template model



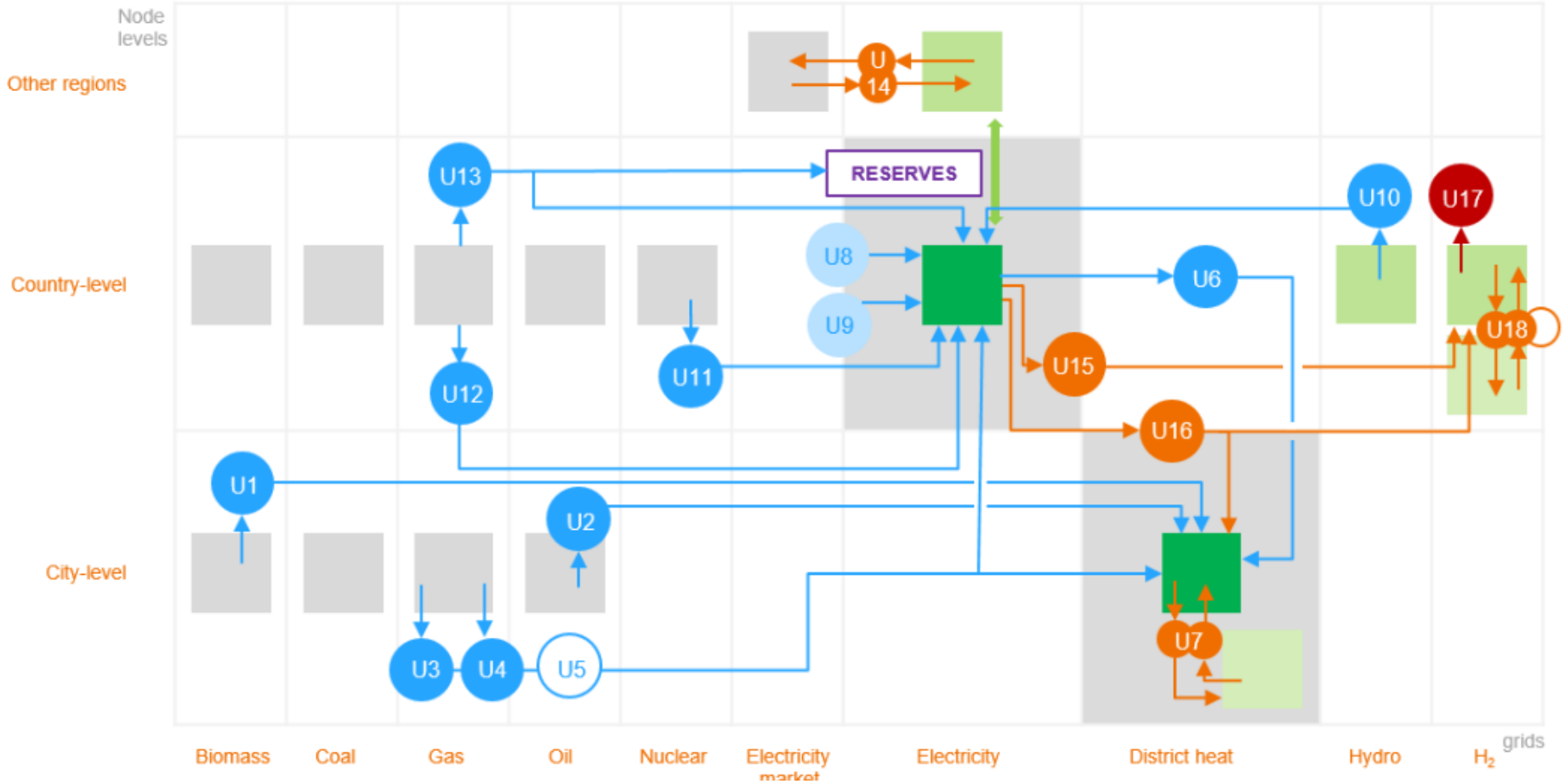
Alternatives include:

- Small battery
- Large battery
- Invest battery
- Irrigation

Scenarios include:

- Small battery without irrigation or investments
- Large battery with irrigation
- Invested battery capacity with irrigation

Representing detail in space and technology



- Node-unit stucture controls level of detail vs. run time
- Typical use case is that nodes are used for both country- and city-level and also as storages or demands

Parameters for constraints

In most cases the users want to give additional constraints that the model has to respect

Most important constraints include

- Unit capacity (MW)
- Energy conversion process with efficiency

Typical additional constraints are

- Minimum reserve capacity (MW at each timestep),
- Minimum inertia limit (MWs at each timestep),
- Max ramp up/down rates for units,
- Maximum/minimum invest on certain technology
- Fixed generation of certain unit

Parameters that control the constraints need to reflect reality

- Too loose constraints can give too optimistic (unrealistic) solutions
- Too strict constraints can lead to too limited (unrealistic) solutions

More model objects

Commodity is a sink or source of energy with price.

Storages are defined to nodes. By default, node maintains energy balance, but adding storage includes a ’state’ variable. The state is stored energy (MWh), while the charging/discharging capacity (MW) is defined as a connection or unit. There are built-in constraints for managing storage state over start/end/gaps in timeline.

Use **equality constraint** e.g. to fix ratios of multiple inputs/outputs, and **greater than constraint** e.g. to fix the operating area of CHP plants.

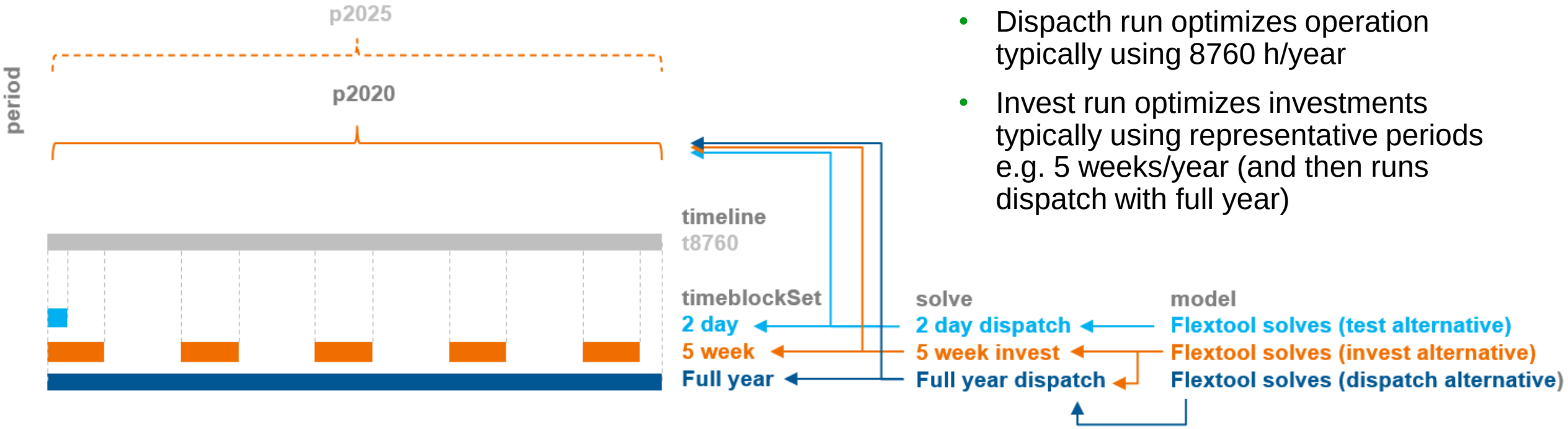
Create **reserves** by fixed time series, dynamically or by largest unit failure.

Commodities can have **emissions**, and emission constraints can be set by creating groups. **Group constaints** can be used also for e.g. inertia or non-syncronous capacity.

Time settings

Temporal building blocks:

- Timeline (t):Timesteps are an ordered series of timesteps, e.g. hours in a year.
- Timeblockset: Choose a block(s) from the timeline to be used in a particular model
- Time period: Annual or other period (needed in multi-year modelling)
- Use solve and model (with alternatives) to choose runs

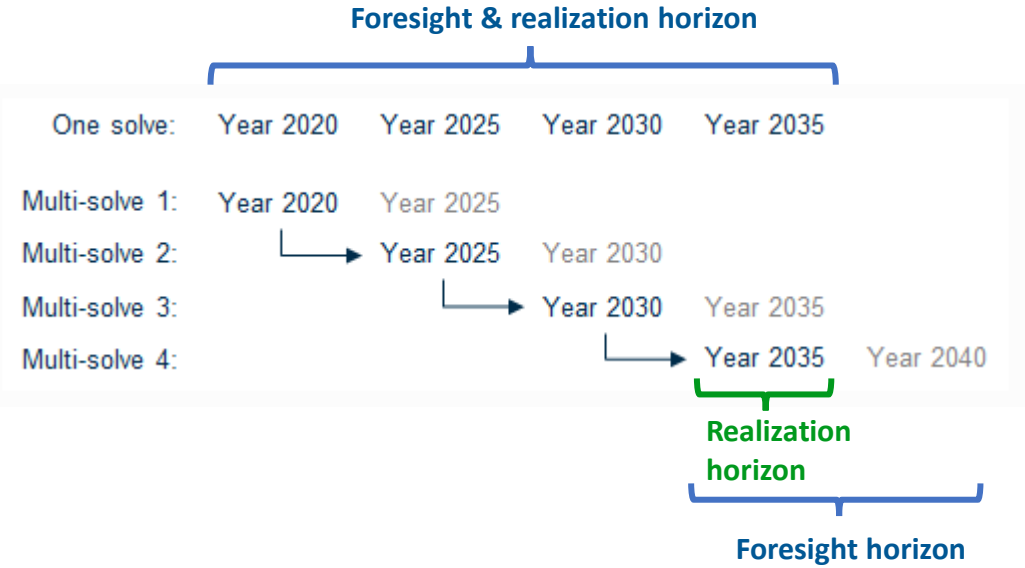


- Dispacth run optimizes operation typically using 8760 h/year
- Invest run optimizes investments typically using representative periods e.g. 5 weeks/year (and then runs dispatch with full year)

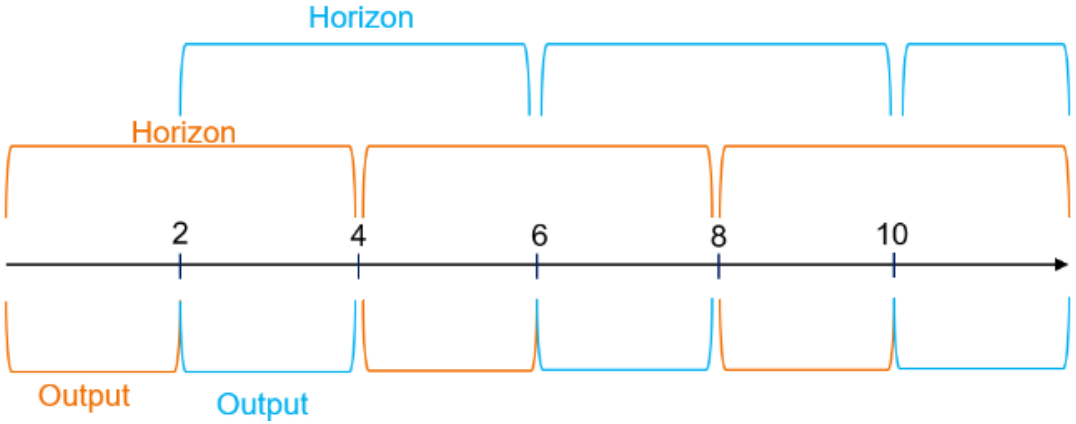
Temporal options: multi-year and rolling

LEAP-RE

1. Multi-year invest sequence
   - Using periods
   - For pathway analysis



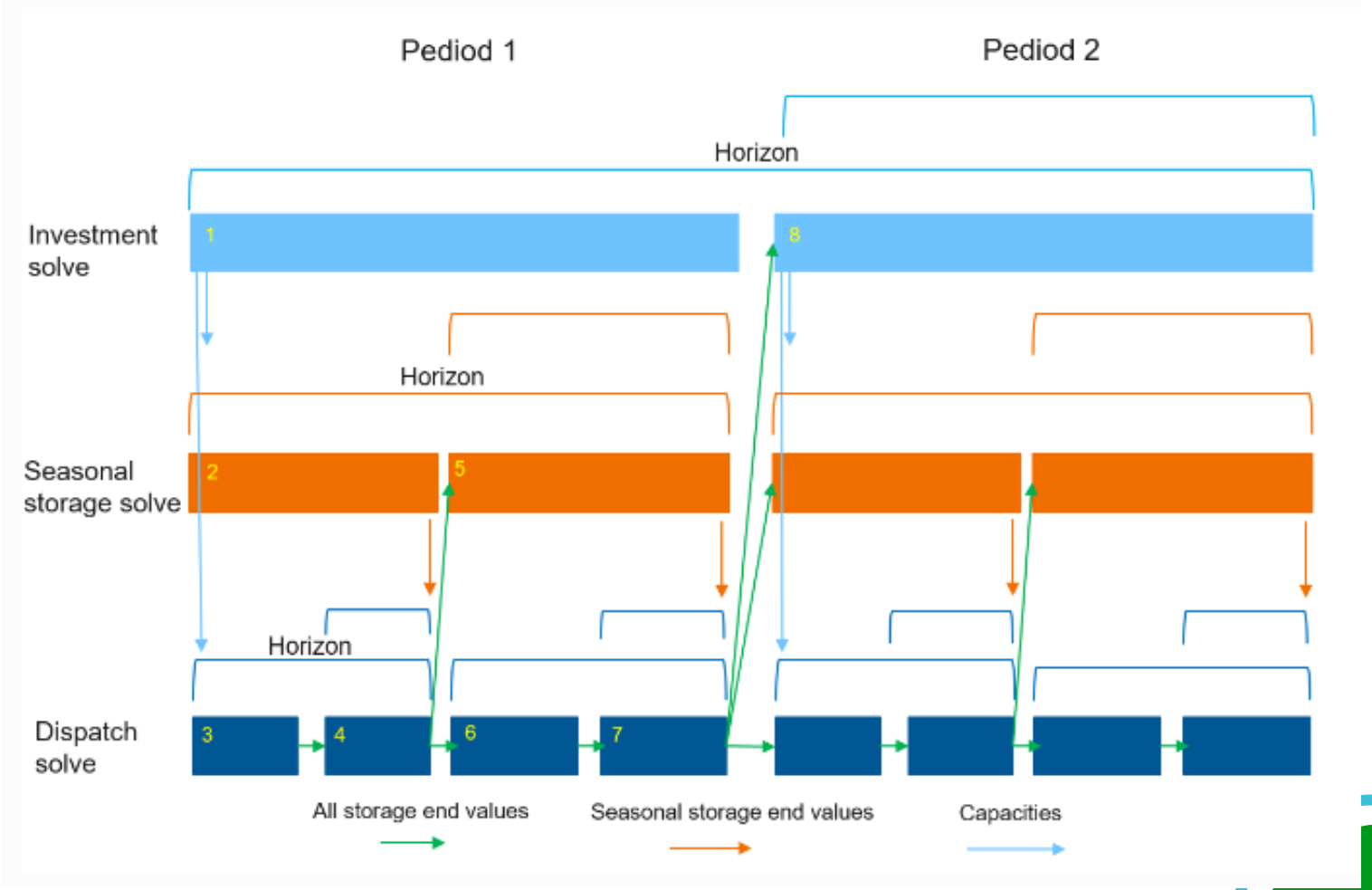
2. Rolling window for dispatch run
   - Using roll settings
   - Eases computation
   - Can create problems for investments and long-term storages



Temporal options: Nested rolling window

LEAP-RE

3. Nested rolling window
   - Reduce computational effort with consideration for investments and long-term storages
   - Solve investments and seasonal storage with reduced resolution using longer timesteps, representative periods or sequenced investments



Strenghts and weaknesses

Strengths

- Free, open source, and relatively simple to use
- Flexible time settings, allows for e.g. rolling horizon, sub-hour resolution and pathway modelling
- Highlights possible operational problems and costs arising from insufficient flexibility
- Capable for investment scenarios to study least cost solutions for the flexibility issues and long-term capacity expansion planning
- Capable for integrated modelling of storages, sector coupling, unit level constraints, etc
- Support for commercial solvers (CPLEX)

Weaknesses

- Deterministic model; does not have forecast errors *
- Simplified power transmission: transport model (MW constraint only) - or any other energy/material flow
- At fully open mode open-source solvers (HiGHS) are slower than commercial

** Apart from what is accounted for in the reserves*

Finally: Spine Toolbox for managing workflows

Spine Toolbox is a Python-based interface to manage data and modelling workflow and acts as a front-end to FlexTool, but also other models.

Allows maintaining and sharing repeatable workflows to supply data to different modelling tools. Toolbox includes tools to manage and transform data, data structures, and data formats.

Toolbox improves the quality and repeatability of the entire modelling process, by offering a structural approach to:

- Several data sources and their updates
- Data transformations between data, model and results
- Up to hundreds of scerarios
- Sharing, documenting and explaining work to others

FlexTool utilizes the Toolbox data structure

Creating scenarios from alternatives (and parallelizing their execution) is a Toolbox functionality applied in FlexTool.

The FlexTool unit-node stucture is one use case of the generic data structure of Toolbox.

In Toolbox, the user can freely define the entity classes and parameters the model uses.