Handling of long-term storage in multi-horizon stochastic programs

Michal Kaut

March 14, 2024

This paper introduces a method for incorporating long-term storage into the multi-horizon modelling paradigm, thereby expanding the scope of problems that this approach can address. The implementation presented here is based on the HyOpt optimization model, but the underlying concepts are designed to be adaptable to other models that utilize the multi-horizon approach.

We demonstrate the effects of several formulations on a case study that explores the electrification of an offshore installation using wind turbines and a hydrogen-based energy storage system. The findings suggest that the formulations offer a realistic modelling of storage capacity, without compromising the advantages of the multi-horizon approach.

Multi-horizon stochastic programming (Kaut et al., 2019) is a modelling framework for reducing size of optimization models with several time scales, typically a combination of strategic (long-term) and operational (short-term) decisions. It works by decoupling the strategic and operational decisions, as described in the next section. This allows for a significant reduction in the size of the generated models and consequently in the required solution time.

The trade-off for this simplification is that the decoupling prevents accurate modelling of long-term storages, such as hydropower with multi-year reservoirs. In this paper, we present formulas for approximation of inventory levels and required storage capacities, for several types of storages and model structures. This is expected to broaden the range of models to which the multi-horizon approach can be applied.

The remainder of the paper is structured as follows: We start by outlining the main aspects of the multi-horizon modelling approach in Section 1 and introducing the relevant components of our 'base' optimization model in Section 2. Following this, we describe how to add long-term storage to the model in Section 3 and demonstrate the proposed approach on a test case in Section 4.

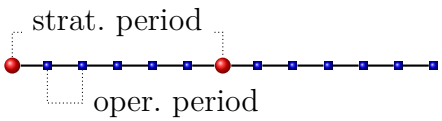
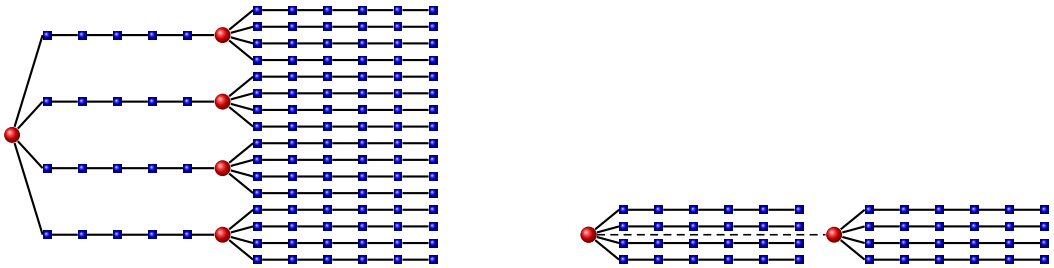


Figure 1: Structure of a simple model with two strategic-decision nodes.



(a) Standard scenario tree **(b)** Multi-horizon scenario tree

Figure 2: Scenario tree with two strategic periods and four operational scenarios in each strategic node. Note that in the multi-horizon tree, the second strategic node is connected to the first strategic node, instead of the preceding operational nodes.

1. Multi-horizon modelling

To understand the concept of the multi-horizon approach, consider an optimization model for designing an energy infrastructure with the objective to minimize the overall costs while meeting a specified energy demand. This necessitates the use of two types of variables in the models: *strategic* decisions related to the infrastructure and *operational* decisions concerning its usage, potentially under different scenarios. We assume that the strategic decisions occur infrequently and refer to the time intervals between them as *strategic periods*, typically measured in months or years. Conversely, the frequency of the operational decisions must be such that it adequately captures the system dynamics, typically ranging from seconds to hours.

This results in a model structure depicted in Fig. 1, with strategic nodes denoted by '●' and operational nodes by '▪'. It is important to note that the sequence of operational nodes would typically be much longer, often extending to thousands of nodes – for instance, one year with hourly time steps equates to 8760 operational nodes.

Suppose we aim to evaluate the infrastructure using four *operational scenarios* instead of one, to obtain a more robust performance estimate. Typically, this would result in a structure depicted in Fig. 2a. It is evident that the model size grows exponentially with the number of periods, quickly leading to intractable models.

The multi-horizon approach circumvents this exponential growth by decoupling the strategic nodes from the operational scenarios, as shown in Fig. 2b. An alternative interpretation of the approach is that we construct a scenario tree for the strategic nodes and interpret the operational nodes as being 'subtrees' of their strategic nodes,

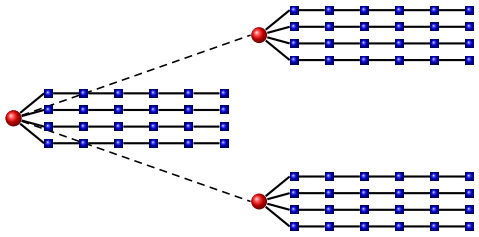


Figure 3: Multi-horizon scenario tree with both strategic uncertainty (modelled by the two strategic nodes in the second strat. period) and operational uncertainty (modelled by operational scenarios attached to all strategic nodes). Dashed lines represent the strategic tree, i.e., the connections between the strategic nodes.

as shown in Fig. 3.[1] In other words, we have two types of scenario structures, each with different time discretizations and horizons. While this approach can theoretically be generalized to more than two levels (hence its name), all papers and models we are aware of utilize only the two levels outlined above.

The multi-horizon approach is particularly well suited for large-scale energy-system models, such as the EMPIRE model (Backe et al., 2022, Skar et al., 2016) and the stochastic version of the TIMES model (Loulou and Lettila, 2016, Ringkjøb et al., 2020, Seljom and Tomasgard, 2015). It has also been applied to natural-gas infrastructure modelling (Hellemo et al., 2013), hydro-power planning (Abgottspon and Andersson, 2016), optimization of building retrofits (Rocha et al., 2016), design of charging infrastructure Bordin and Tomasgard (2019), power-system restructuring and capacity expansion (Backe et al., 2021, Bordin et al., 2021), and modelling of offshore energy hubs Zhang et al. (2022).

In addition to applications, Maggioni et al. (2019) explores the theoretical properties of multi-horizon stochastic programs, including bounds on the significance of stochasticity in these models. Moreover, Zhang et al. (2023a,b) develop several decomposition-based solution techniques that take advantage of the hierarchical structure of multi-horizon models. They achieved a significant reduction in solution times compared to generic decomposition methods.

1.1. Reducing the length of operational scenarios

Once we have separated the operational scenarios from different strategic nodes, we can further decrease the model size by reducing the length of the operational scenarios. For instance, instead of one yearly scenario, we can represent the year by one or more *representative days* or weeks and then scale their effects (costs etc.) to one year. With long

[1] Some authors refer to the resulting model as *two-stage*, with the first stage consisting of the strategic variables and the second stage consisting of the operational variables and being conditional on the first stage. Note that this interpretation is independent on the number of strategic periods in the model.

strategic periods, this can reduce the model size by more than an order of magnitude, making this approach appealing for large-scale and long-term models. Notably, both the aforementioned energy-system models, EMPIRE and TIMES use this approach.

1.2. Long-term storage problem

The size-reduction comes at a cost: since we have eliminated the link between operational scenarios in consecutive strategic periods, the approach does not provide a natural way of modelling of storages spanning several strategic periods. This becomes even more complex if we use multiple representative periods instead of operational scenarios spanning the entire duration of the strategic periods. Strømholm and Rolfsen (2021) partially addressed this issue by providing formulas for the inventory level at the end of strategic periods, but their approach does not consider how this affects the required storage capacities.

This means that it has so far been impossible to use a multi-horizon model for dimensioning long-term storages – one can include them in the model, but their capacities would be adjusted for the representative periods in use (days or weeks), not for the period they are meant to represent (year).

In this paper, we address this issue and demonstrate how to approximate storage capacities and inventory levels for several types of operational scenarios and time scopes. For this, we utilize the HyOpt optimization model (Kaut et al., 2019), which is presented in the next section.

2. Relevant aspects of the HyOpt model

HyOpt is a model for optimal dimensioning of energy systems, implementing the concepts described in Section 1, with a scenario tree as in Fig. 3, i.e., a tree consisting of strategic-decisions nodes, each with an attached set of operational scenarios.

Given that HyOpt is a complex model, it is not feasible to present a complete formulation here. For this, we direct the reader to Kaut et al. (2019), or to our implementation in the FICO™ Mosel optimization language, freely available from `https://gitlab.sintef.no/open-hyopt`. Instead, this section presents a subsection of the model relevant to storage modelling.

2.1. Notation

Indices

sn	strategic node, i.e., one node of the strategic scenario tree
sc	operational scenario
op	operational period

Table 1: Weights and multipliers of 3 operational scenarios in a one-year strategic period ($\Delta T^{\mathrm{SP}}_{sn} = 365$ days). $\Delta T^{\mathrm{SC}}_{sc}$ is given in days. The last column shows how many days each scenario represents: days $= \Delta T^{\mathrm{SC}}_{sc} \times M^{\mathrm{SC}}_{sn,sc}$.

oper. sc.	$W^{\mathrm{SC}}_{sn,sc}$	$\Delta T^{\mathrm{SC}}_{sc}$	$T^{\mathrm{M}}_{sn,sc}$	$M^{\mathrm{SC}}_{sn,sc}$	days
scen. 1	182/365	7	365/7	26	182
scen. 2	182/365	7	365/7	26	182
scen. 3	1/365	1	365	1	1
sum	1.0				365

Parameters

$W^{\mathrm{SC}}_{sn,sc}$	weight of oper. scenario *sc* in strategic node *sn*
$T^{\mathrm{M}}_{sn,sc}$	time-scaling multiplier for scenario *sc* in strat. node *sn*
$M^{\mathrm{SC}}_{sn,sc}$	overall multiplier for scenario *sc* in strat. node *sn*
$\Delta T^{\mathrm{SP}}_{sn}$	duration of strategic period in strat. node *sn*
$\Delta T^{\mathrm{SC}}_{sc}$	duration of operational scenario *sc*
$\Delta T^{\mathrm{OP}}_{sc,op}$	duration of oper. period *op* in operational scenario *sc*

2.2. Scenario-tree structure in HyOpt

In HyOpt, the strategic scenario tree consists of *strategic nodes*, each belonging to one *strategic period*. A *strategic scenario* is a path from the *root* of the strategic tree to one of the leaves.

Each of the strategic nodes has one or more *operational scenarios* associated with it. These scenarios can vary in length and time resolution. Each scenario also has an assigned *weight* $W^{\mathrm{SC}}_{sn,sc}$, denoting the time spent in the scenario as a fraction of the strategic period – implying $\sum_{sc} W^{\mathrm{SC}}_{sn,sc} = 1$ for all strategic nodes *sn*.

If the duration of an operational scenario does not match the duration of its associated strategic period, we must scale all effects of the scenario up to the strategic period using the multiplier

$$M^{\mathrm{SC}}_{sn,sc} = W^{\mathrm{SC}}_{sn,sc} \times T^{\mathrm{M}}_{sn,sc} \tag{1}$$

where

$$T^{\mathrm{M}}_{sn,sc} = \Delta T^{\mathrm{SP}}_{sn} / \Delta T^{\mathrm{SC}}_{sc} \,. \tag{2}$$

As an example, Table 1 shows a situation where a strategic node in a one-year strategic period has three operational scenarios, two weekly for normal situations and one daily representing an extreme day. There, the last column shows how may days of the year are represented by each scenario. The table also illustrates that the *weights* do *not* equate to probabilities if the scenarios differ in length: the extreme day occurs (on average) once per year while each of the normal scenarios occurs 26 times, so the probabilities of these scenarios occurring are 1/53 and 26/53, respectively.

3. Storage handling

This section presents formulas for managing storages in the HyOpt model. In particular, we demonstrate how to adjust the storage capacity, dependent on the duration of the operational scenarios and the time-scale of the storages. We also introduce a new concept, *scenario groups*, and demonstrate its impact on the presented formulas.

3.1. Relevant notation

Indices

i storage node, i.e., node in the network modelling the system

Variables

$\mathbf{inv}_{i,sn,sc,op}$	inventory level of storage i at the end of oper. period (sn, sc, op)
$\mathbf{inv}^{\text{init}}_{i,sn,sc}$	inventory level at the start of oper. scenario (sn, sc)
$\mathbf{inv}^{\text{end}}_{i,sn,sc}$	inventory level at the end of oper. scenario (sn, sc)
$\mathbf{inv}^{\text{init}}_{i,sn}$	initial inventory level of storage i at strat. node sn
$\mathbf{inv}^{\text{end}}_{i,sn}$	final inventory level of storage i at strat. node sn

All the inventory variables are non-negative and limited by $\mathbf{cap}_{s,sn}$, the capacity available at strat. node sn:

$$0 \leq \mathbf{inv}_{i,sn,sc,op} \leq \mathbf{cap}_{i,sn}\,, \tag{3}$$

and correspondingly for the other variables.

3.2. Scaling storages from operational scenarios

While costs can be scaled by $M^{\text{SC}}_{sn,sc}$, managing storage needs additional consideration. At least two issues need attention: the inventory level at the end of a strategic period, and the minimum storage capacity required for the presented scenarios.

If we denote by $\mathbf{inv}^{\Delta}_{i,sn,sc}$ the change of the inventory during oper. scenario sc,

$$\mathbf{inv}^{\Delta}_{i,sn,sc} = \mathbf{inv}^{\text{end}}_{i,sn,sc} - \mathbf{inv}^{\text{init}}_{i,sn,sc}\,,$$

then the overall change in the inventory at strategic node sn is

$$\mathbf{inv}^{\Delta}_{i,sn} = \sum_{sc} M^{\text{SC}}_{sn,sc} \times \mathbf{inv}^{\Delta}_{i,sn,sc}\,.$$

Assuming further that all operational scenarios begin with the same inventory level,

$$\mathbf{inv}^{\text{init}}_{i,sn,sc} = \mathbf{inv}^{\text{init}}_{i,sn}\,,$$

the final inventory is given by (Strømholm and Rolfsen, 2021)

$$\mathbf{inv}^{\text{end}}_{i,sn} = \mathbf{inv}^{\text{init}}_{i,sn} + \mathbf{inv}^{\Delta}_{i,sn}\,. \tag{4}$$

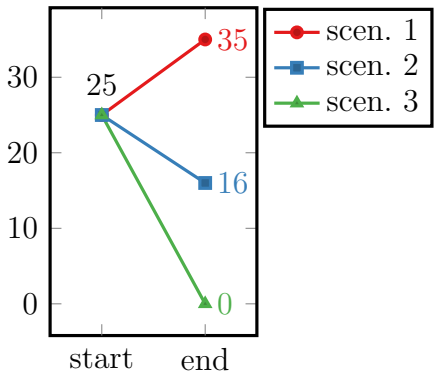


Figure 4: Storage level changes with common starting point

The aforementioned requirement that all operational scenarios within a single strategic node begin with the same inventory level corresponds to the interpretation of operational scenarios as a set of possible random occurrences: since we do not know which scenario will happen, we cannot adjust inventory levels in anticipation.

This requirement also assists in determining the storage dimensions. To illustrate this, consider the three scenarios from Table 1 and assume that the inventory changes of storage i in the three scenarios are (10, -9, -25), respectively. With a cost associated to the installed storage capacity and no additional constraints, the model will choose the minimum capacity that satisfies Eq. (3) in all scenarios. In our case, this leads to the inventory levels presented in Fig. 4, requiring storage capacity of 35[2].

However, it is important to note that this storage capacity is an underestimation. Considering that both 'normal' scenarios (scen. 1 and scen. 2) occur 26 times during the strategic period and assuming they occur randomly, we can expect multiple consecutive occurrences of each. This significantly increases the required capacity.

For instance, starting with two consecutive occurrences of scen. 1 results with an inventory level of $25 + 2 \times 10 = 45$, requiring $\mathbf{cap}_{i,sn} \geq 45$. Simultaneously, a single occurrence of scen. 2 preceded or followed by scen. 3 would lead to an infeasible inventory level $25 - 25 - 9 = -9$. To avoid this infeasibility, the initial inventory level would need to increase to $25 + 9 = 34$, requiring $\mathbf{cap}_{i,sn} \geq 44$ ...and $\mathbf{cap}_{i,sn} \geq 45 + 9 = 54$ if we wanted to consider the first sequence as well. In both cases, allowing more repetitions would increase the required capacity even further.

Conversely, if we were certain that the two normal scenarios alternate, then no repetitions could occur and $\mathbf{cap}_{i,sn} = 44$ would be sufficient to cover all allowed permutations. This shows that the degree of capacity underestimation is inherently case-dependent.

3.3. Operational scenarios in sequence

In the previous sections, we assumed that operational scenarios occur randomly. However, this assumption does not hold when the scenarios represent sequential events, such

[2]This assumes that the inventory levels change monotonously from the start to the end of each operational period, which is typically not the case. We will address this issue later.

Table 2: Weights and multipliers of 5 operational scenarios. Again, $\Delta T^{\mathrm{d}}_{sc}$ is the duration of oper. scenario sc in days and column 'days' shows how many days each scenario represents. In addition, the last column shows the inventory changes scaled to the whole strategic period, $\mathbf{inv}^{\Delta\mathrm{SP}}_{i,sn,sc} = T^{\mathrm{M}}_{sn,sc} \times \mathbf{inv}^{\Delta}_{i,sn,sc}$.

oper. sc.	$W^{\mathrm{SC}}_{sn,sc}$	$\Delta T^{\mathrm{d}}_{sc}$	$T^{\mathrm{M}}_{sn,sc}$	$M^{\mathrm{SC}}_{sn,sc}$	days	$\mathbf{inv}^{\Delta}_{i,sn,sc}$	$\mathbf{inv}^{\Delta\mathrm{SP}}_{i,sn,sc}$
winter	91/365	7	365/7	13	91	-10	-130
spring	91/365	7	365/7	13	91	15	195
summer	91/365	7	365/7	13	91	-5	-65
autumn	91/365	7	365/7	13	91	0	0
bad day	1/365	1	365	1	1	-5	-5
sum	1.0				365		-5

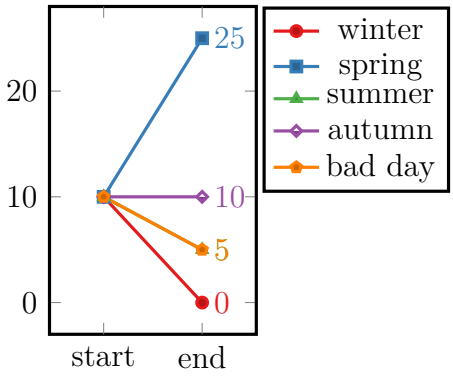


Figure 5: Storage level changes with oper. scenarios from Table 2 interpreted as random events

as seasons within a year. This is frequently employed in long-term models, as shown in Skar et al. (2016) or Strømholm and Rolfsen (2021).

In these instances, the scenarios do *not* occur randomly throughout the strategic period. Rather, if we have one scenario per season, its probability is 100% during that season and 0% elsewhere. It follows that the scenarios occur consecutively – 13 times in the case of weekly scenarios representing seasons. This significantly impacts the necessary storage capacity.

To demonstrate this, consider the scenarios from Table 2. If we treat these as random events, the overall storage change is -5 and the required capacity is 25, as shown in Fig. 5. To enable the scenarios to represent the sequence of seasons, we first need to decide how to manage the 'bad day' scenario. We have allocated it to spring, which is thus represented by two scenarios with a total inventory change of 190. The inventory-level changes for each scenario are displayed in the last column of Table 2 and the entire annual profile in Fig. 6. The figure shows that the minimal storage capacity required in this case is 195, nearly eight times more than in the original approach. Note that the total inventory-level change remains at -5, unaffected by the grouping.

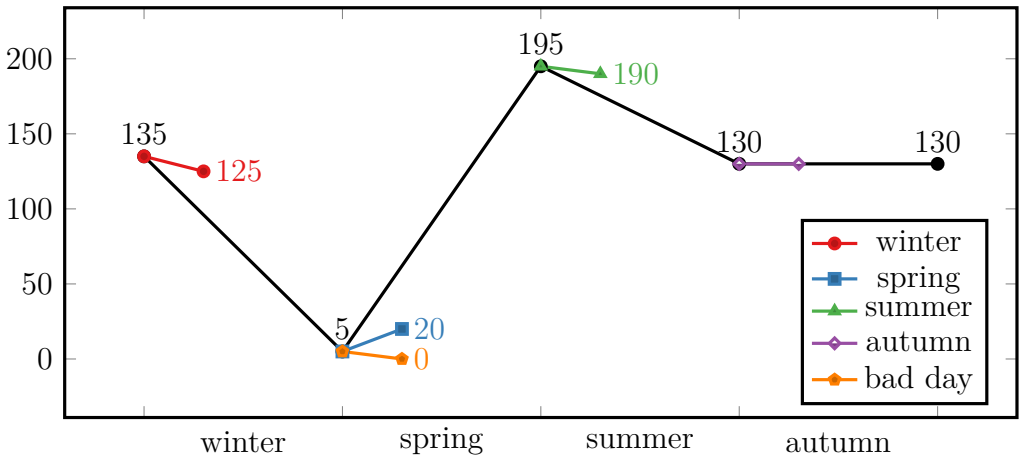


Figure 6: Storage level changes with oper. scenarios from Table 2, interpreted as four seasons in sequence with the 'bad day' scenario assigned to spring.

3.3.1. Extra model notation

To implement the scenario groups in the model, we need to introduce extra notation:

Parameters and sets

$\boldsymbol{G}^{\mathrm{SC}}_{sn}$	ordered list of scenario groups at strategic node sn
$\boldsymbol{S}^{\mathrm{G}}_{sn,g}$	set of operational scenarios in group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$\Delta T^{\mathrm{G}}_{sn,g}$	duration of scenario group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$W^{\mathrm{SC,G}}_{sn,g,sc}$	relative weight of scenario sc within scen. group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$T^{\mathrm{M,G}}_{sn,g,sc}$	time scaling factor for scenario sc within scen. group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$M^{\mathrm{SC,G}}_{sn,g,sc}$	multiplier for scenario sc within scen. group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$\boldsymbol{g}^{\mathrm{first}}_{sn}$	first scen. group in strat. node sn
$\boldsymbol{g}^{\mathrm{last}}_{sn}$	last scen. group in strat. node sn

Variables

$\mathbf{inv}^{\mathrm{G,init}}_{i,sn,g}$	inventory level at the start of scenario group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$
$\mathbf{inv}^{\mathrm{G,end}}_{i,sn,g}$	inventory level at the end of scenario group $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$

The inventory-level variables are again non-negative and limited by $\mathbf{cap}_{i,sn}$. In addition, the new entities are connected by the following relations:

Table 3: Alternative scenarios for the summer group. Analogously to Table 2, $\mathbf{inv}^{\Delta\mathrm{G}}_{i,sn,g,sc} = T^{\mathrm{M,G}}_{sn,g,sc} \times \mathbf{inv}^{\Delta}_{i,sn,sc}$ is the inventory-level change during the scenario, scaled to the duration of the group. The last column shows the maximum inventory level reached within each oper. scenario, relative to its starting level, $\mathbf{inv}^{\max\Delta}_{i,sn,sc} = \max_{op} \mathbf{inv}_{i,sn,sc,op} - \mathbf{inv}^{\mathrm{init}}_{i,sn,sc}$.

oper. scen.	$W^{\mathrm{SC}}_{sn,sc}$	$W^{\mathrm{SC,G}}_{sn,g,sc}$	$T^{\mathrm{M,G}}_{sn,g,sc}$	$M^{\mathrm{SC,G}}_{sn,g,sc}$	$\mathbf{inv}^{\Delta}_{i,sn,sc}$	$\mathbf{inv}^{\Delta\mathrm{G}}_{i,sn,g,sc}$	$\mathbf{inv}^{\max\Delta}_{i,sn,sc}$
summer-1	42/365	6/13	13	6	5	30	10
summer-2	42/365	6/13	13	6	−12	−72	0
summer-3	7/365	1/13	13	1	−23	−23	0

$$
\begin{aligned}
W^{\mathrm{SC,G}}_{sn,g,sc} &= W^{\mathrm{SC}}_{sn,sc} \,/ \sum_{\tilde{sc}\in g} W^{\mathrm{SC}}_{sn,\tilde{sc}} \\
T^{\mathrm{M,G}}_{sn,g,sc} &= \Delta T^{\mathrm{G}}_{sn,g} / \Delta T^{\mathrm{OP}}_{sc,op} \\
M^{\mathrm{SC,G}}_{sn,g,sc} &= W^{\mathrm{SC}}_{sn,sc} \times T^{\mathrm{M,G}}_{sn,g,sc} \\
\mathbf{inv}^{\mathrm{G,init}}_{i,sn,\boldsymbol{g}^{\mathrm{first}}_{sn}} &= \mathbf{inv}^{\mathrm{init}}_{i,sn} \\
\mathbf{inv}^{\mathrm{init}}_{i,sn,sc} &= \mathbf{inv}^{\mathrm{G,init}}_{i,sn,g} \\
\mathbf{inv}^{\mathrm{G,end}}_{i,sn,g} &= \mathbf{inv}^{\mathrm{G,init}}_{i,sn,g} + \sum_{sc\in \boldsymbol{S}^{\mathrm{G}}_{sn,g}} M^{\mathrm{SC,G}}_{sn,g,sc} \times \mathbf{inv}^{\Delta}_{i,sn,sc} \\
\mathbf{inv}^{\mathrm{G,init}}_{i,sn,g} &= \mathbf{inv}^{\mathrm{G,end}}_{i,sn,g-1} \text{ if } g-1 \in \boldsymbol{G}^{\mathrm{SC}}_{sn} \text{ else } \mathbf{inv}^{\mathrm{init}}_{i,sn,sn} \\
\mathbf{inv}^{\mathrm{end}}_{i,sn} &= \mathbf{inv}^{\mathrm{G,end}}_{i,sn,\boldsymbol{g}^{\mathrm{last}}_{sn}}
\end{aligned}
$$

where $g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}$ and $sc \in \boldsymbol{S}^{\mathrm{G}}_{sn,g}$, in all constraints where they appear. Since we treat scenarios within each group as random events, we force them to start from a common initial level. The last constraint ensures that the installed capacity is big enough to handle also the final inventory level in the sequence.

3.3.2. Repeated scenarios

Now, consider the consequences of substituting the current summer scenario with the three weekly scenarios presented in Table 3. There, the last column includes the maximum inventory level during the scenario, relative to its initial level. This implies that the inventory in scenario 'summer-1' initially increases by 10 units and subsequently decreases, resulting in an overall change of +5 units.

Since $30 - 72 - 23 = -65$, the total inventory change during summer remains unchanged, so the overall inventory profile stays the same as in Fig. 6 – except that scenario 'summer-1' increases the required storage capacity from 195 to 205, to accommodate the maximum level reached during this scenario.

However, if this scenario were to occur consecutively two or more times, it would necessitate an even higher capacity. This implies that we need to determine the number of repetitions we wish to consider. The maximum number of repetitions is $M^{\mathrm{SC,G}}_{sn,g,sc}$; for 'summer-1', this translates to 6 consecutive occurrences – but this happens with a probability of only $(W^{\mathrm{SC,G}}_{sn,g,sc})^6 = (6/13)^6 = 0.97\%$, so considering it might be overly conservative. Instead, we have opted to set a limit on the probability we intend to consider, denoted by P^{R}. This restricts the number of consecutive occurrences to

$$seq^{\mathrm{G}}_{sn,g,sc} = \lfloor \ln(P^{\mathrm{R}}) / \ln(W_{sn,s}/W^{\mathrm{g}}_{sn,g}) \rfloor \,, \tag{5}$$

bounded by 1 from below and $M^{\mathrm{SC}}_{sn,sc}$ from above.

In our case, we use $P^{\mathrm{R}} = 5\%$. For 'summer-1', this means considering up to $seq^{\mathrm{G}}_{sn,g,sc} = \lfloor \ln(0.05)/\ln(6/13) \rfloor = 3$ consecutive occurrences. In the initial one, the inventory level goes from 195 to 200, with a maximum at 205. The first repetition starts with inventory level of $195 + 5 = 200$ and the second with $195 + 2 \times 5 = 205$, with a maximum reaching $205 + 2 \times 5 = 215$. This would thus become the new required capacity.

To incorporate this capacity requirement into the model, we merely need to address the final repetition, since the inventory levels change linearly throughout the sequence and the initial occurrence is already accounted for in the model and therefore satisfies Eq. (3). The last occurrence has inventory levels shifted by $(seq^{\mathrm{G}}_{sn,g,sc} - 1)\,\mathbf{inv}^{\Delta}_{i,sn,sc}$, so its version of Eq. (3) is

$$0 \leq \mathbf{inv}_{i,sn,sc,op} + (seq^{\mathrm{G}}_{sn,g,sc} - 1)\,\mathbf{inv}^{\Delta}_{i,sn,sc} \leq \mathbf{cap}_{i,sn} \,, \tag{6}$$

for all operational scenarios sc with $seq^{\mathrm{G}}_{sn,g,sc} > 1$.

3.3.3. Multi-year scenarios

Up until now, we have implicitly assumed that the strategic periods are annual, with the sequence of scenario groups (seasons) spanning the entire strategic period. If we extend the duration to two years, the overall storage-level change doubles to -10 (-5 each year), but this is clearly not the case for inventory levels. In fact, if the overall change in storage level were zero, the second year would mirror first, resulting in no change to the required storage capacity.

This can be addressed analogously to the scenario repetition, using

$$0 \leq \mathbf{inv}_{i,sn,sc,op} + (\lfloor (\Delta T^{\mathrm{SP}}_{sn} / \sum_{g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}} \Delta T^{\mathrm{G}}_{sn,g} \rfloor - 1)\,\mathbf{inv}^{\Delta \mathrm{G}}_{i,sn} \leq \mathbf{cap}_{i,sn} \,, \tag{7}$$

where $\mathbf{inv}^{\Delta \mathrm{G}}_{i,sn}$ is the total inventory-level change in all the scenario groups,

$$\mathbf{inv}^{\Delta \mathrm{G}}_{i,sn} = \sum_{g \in \boldsymbol{G}^{\mathrm{SC}}_{sn}} \sum_{sc \in \boldsymbol{S}^{\mathrm{G}}_{sn,g}} \mathbf{inv}^{\Delta \mathrm{G}}_{i,sn,g,sc} \,.$$

We also need constraints that combine Eqs. (6) and (7), to accommodate repeated scenarios within repeated scenarios groups:

$$0 \leq \mathbf{inv}_{i,sn,sc,op} + (seq^{\mathrm{G}}_{sn,g,sc} - 1)\,\mathbf{inv}^{\Delta}_{i,sn,sc} + (\lfloor \Delta T^{\mathrm{SP}}_{sn} / \Delta T^{\mathrm{G}}_{sn,g} \rfloor - 1)\,\mathbf{inv}^{\Delta}_{i,sn} \leq \mathbf{cap}_{i,sn} \tag{8}$$

Note that Eq. (8) alone is not sufficient, and we need all Eqs. (6) to (8). To demonstrate this, assume that we have a scenario with a storage-level change of 10 and $seq^{\mathrm{G}}_{sn,g,sc}=4$, in a two-year strategic period with an overall inventory-level change of −30. Then

$$(seq^{\mathrm{G}}_{sn,g,sc} - 1)\,\mathbf{inv}^{\Delta}_{i,sn,sc} + (\lfloor \Delta T^{\mathrm{SP}}_{sn} / \Delta T^{\mathrm{G}}_{sn,g} \rfloor - 1)\,\mathbf{inv}^{\Delta}_{i,sn} = 3 \times 10 + 1 \times -30 = 0\,,$$

so Eq. (8) would not have no impact, but Eqs. (6) and (7) would still be necessary.

3.4. Initial and final inventory levels

In the absence of any constraints on initial and final inventory levels, the model is likely to opt for starting each operational scenario with full storage and ending it with an empty one, effectively gaining full storage at no cost. There are two common strategies to tackle this issue: assigning a value to the inventory or requiring that the storage ends with the same level as it started. In HyOpt, we use the latter approach with several *time scopes* for the storage-level looping, to distinguish between short- and long-term storages:

oper-scenario, where we require that the inventory level loops back in every operational scenario, i.e., $\mathbf{inv}^{\Delta}_{i,sn,sc} = 0$. This is typically used for batteries, where each scenario constitutes a daily or weekly 'schedule' with charging overnight.

scen-group, where we mandate that the inventory level loops back within each scenario group, i.e., $\mathbf{inv}^{\Delta \mathrm{G}}_{i,sn,g} = 0$. If the scenario groups represent seasons, this would suit storages that cycle in weeks or a few months, such as medium-sized hydrogen-storage systems or smaller hydro reservoirs.

strat-node, where we require that the inventory level at the end of a strategic node equals the initial level, i.e., $\mathbf{inv}^{\Delta}_{i,sn} = 0$. With yearly strategic time periods, this corresponds to inter-seasonal storage like large hydrogen storage or mid-sized hydro reservoirs.

overall, where we mandate that the final inventory level at the end of all strategic nodes in the last strategic periods finishes on a level that equals the initial level in the first strategic period. This corresponds to large hydro reservoirs.

Note that a storage node with scope 'strat-node' or shorter would not, by definition, require Eqs. (7) and (8), and a storage node with 'oper-scenario' scope would not need Eq. (6) either.

The complete HyOpt implementation of the storage modelling, in the FICO™ Mosel modelling language, can be found in Appendix A.

4. Test case

We consider a small test case inspired by the LowEmission project[3], which involves the electrification of an offshore installation using wind turbines combined with energy

[3]See https://www.sintef.no/projectweb/lowemission/

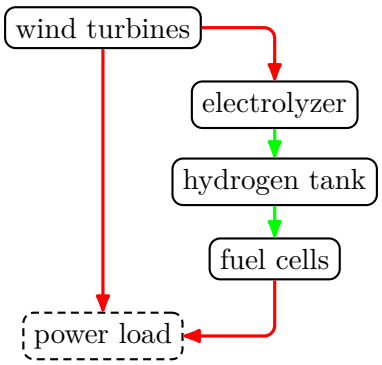


Figure 7: HyOpt representation of of the test case from Section 4. Red arrows represent flow of power, green arrows flow of hydrogen. Nodes whose capacities are being optimized are denoted by solid-line borders.

Table 4: Model time structures considered in the test case. $T^{\mathrm{hor,y}}$ is the overall time horizon, in years, $|\mathcal{T}^{\mathrm{SP}}|$ is the number of strategic periods, and $\Delta T^{\mathrm{SP,y}}_{sn}$ shows the duration of the strat. periods, in years.

struct.	$T^{\mathrm{hor,y}}$	$\lvert\mathcal{T}^{\mathrm{SP}}\rvert$	$\Delta T^{\mathrm{SP,y}}_{sn}$	wind-data year
$\mathbf{1 \times 1}$	1	1	1	2018
$\mathbf{1 \times 5}$	5	1	5	2018–2022
$\mathbf{5 \times 1}$	5	5	$(1, 1, 1, 1, 1)$	2018–2022

storage. In our case, the storage system is hydrogen-based, comprising an electrolyzer, a hydrogen tank, and fuel cells. In HyOpt, this results in a network structure presented in Fig. 7.

The wind-production data are derived from actual wind-speed measurements from the Ekofisk field in the North Sea in years 2018–2022. These measurements, obtained from the Norwegian Meteorological Institute using its Frost API[4], are converted into a wind-production capacity factor using a production profile for Vestas *V164/8000* wind turbine, sourced from the Open Energy Platform[5]. We assume a constant power load of 20 MW. All cost and performance data are obtained from an open HyOpt test case, accessible at `https://gitlab.sintef.no/open-hyopt/test-case-1`.

4.1. Case variants

We evaluate the model using three time structures, 1×1, 1×5, and 5×1, presented in Table 4. The $a \times b$ notation represents a strategic periods, each lasting b years. For each time structure, we examine the impact of the following operational variants:

full, with one operational scenario per strategic period, spanning the entire length of the period.

[4]See `https://frost.met.no/`.
[5]See `https://openenergy-platform.org/`.

Table 5: Number of weeks in operational scenarios and the size reduction compared to the full model, for the tested time structures and model variants

	1×1		1×5		5×1	
case variant	weeks	reduct.	weeks	reduct.	weeks	reduct.
full	52	–	260	–	260	–
mean	4	13	4	65	20	13
mean+min	8	6.5	8	32.5	40	6.5

mean, with four operational scenarios per strategic period, each lasting one week. Each scenario represents a season and is chosen as the week in the data whose average capacity factor is closest to the seasonal average within the given time interval.[6]

mean+min, with eight weekly operational scenarios per strategic period. For each season, we select the week with the smallest above-average capacity factor, and the week with the smallest capacity factor as the worst case scenario. The scenario weights are chosen so that the weighted average is equal to the average capacity factor of the season.

In all cases, the operational scenarios use an hourly time resolution. We use astronomical definitions of seasons, where each season lasts three months and winter starts on December 1. As a result, our 'year' runs from December to November to ensure it consists of four complete seasons. In other words, when we refer to, for example, 2018, the actual interval is from December 1, 2017, to November 30, 2018.

The sizes of the scenario trees for the variants are summarized in Table 5. There, we observe that the size reduction from using the weekly scenarios ranges from 6.5 times to 32.5 times, compared to the case with a full time sequence.

To examine the effect of scenario groups, as described in Section 3.3, we use two versions of the multi-scenario variants ('mean' and 'mean+min'):

fan, where all scenarios are interpreted as random events, resulting in a single 'fan' of scenarios.

groups, with scenarios grouped by season (one per season for 'mean' and two for 'mean+min').

This results in a total of five case variants for testing.

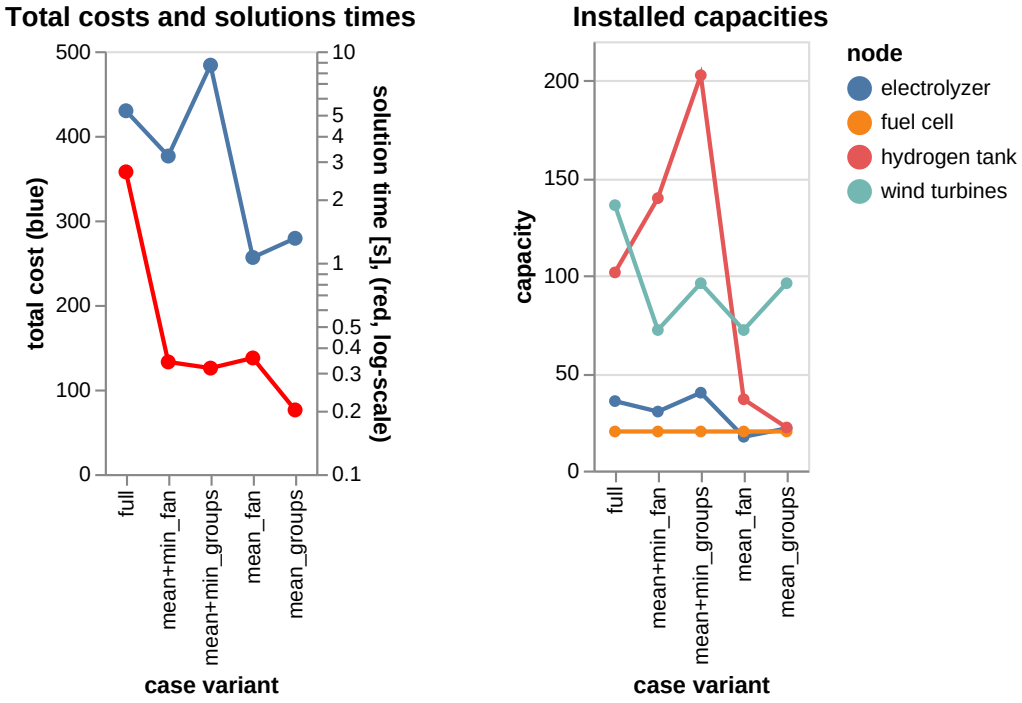


Figure 8: Results for time struct. '1 × 1': one strat. period one year long

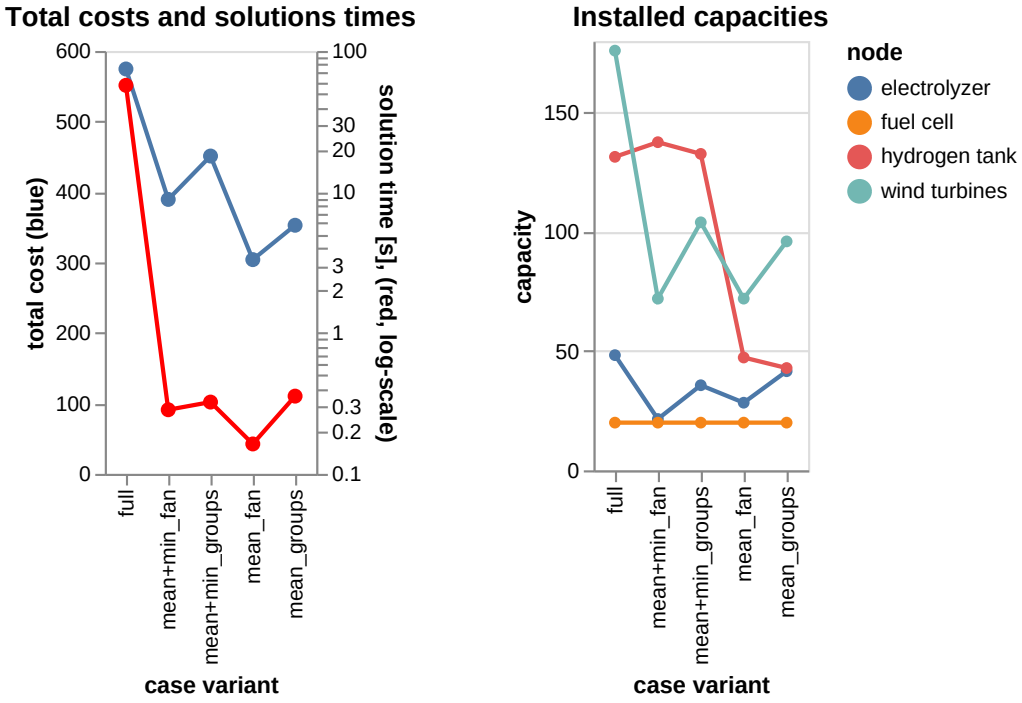


Figure 9: Results for time struct. '1 × 5': one strat. period 5 years long

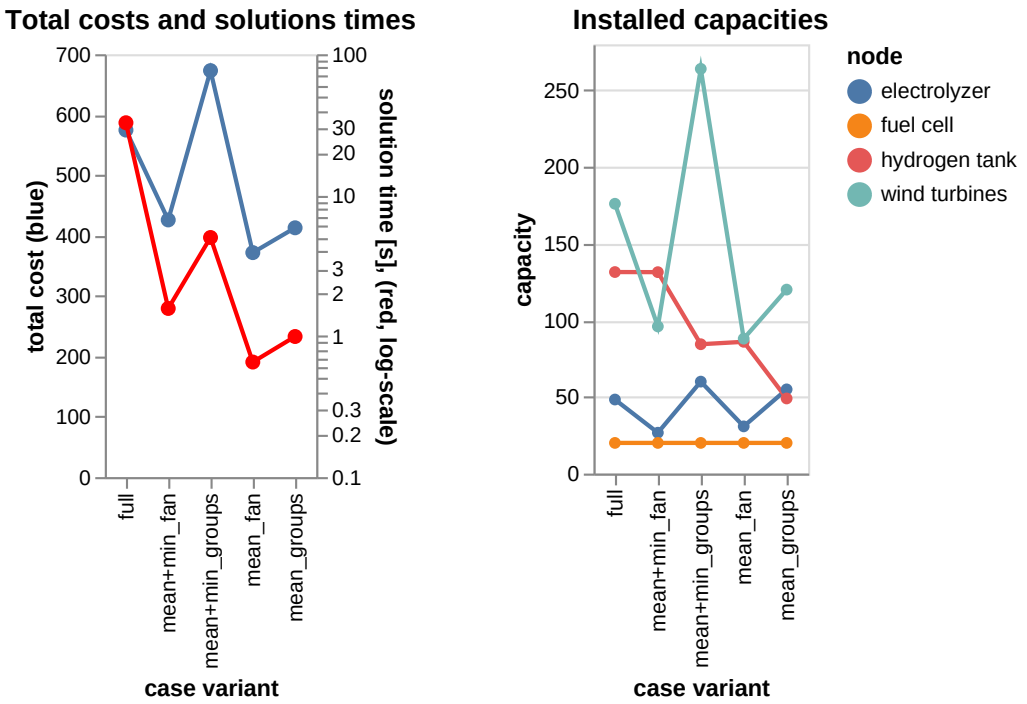


Figure 10: Results for time struct. '5×1': 5 yearly strat. periods

4.2. Results

All test variants were solved using FICO™ Xpress Solver v 8.14, on a laptop with Intel® Core™ i7-7600U CPU operating at 2.80 GHz, and 16 GB of RAM. The results for the one-year case are displayed in Figs. 8 to 10. There, the left panels confirm that the solution times for the scenario-based versions are significantly shorter than for the 'full' model, with a speed-up corresponding to the problem-size reduction shown in Table 5. Specifically, the speed-up ranges from 5–6 in the worst case of the '5×1' variant to over 100 for the '1×5' variant.

In terms of the objective function, i.e., the total costs, we can make the following observations:

- The 'mean+min' variants are more costly than 'mean'. This is expected, as they must account for the extreme scenarios.
- The 'groups' variants are more costly than 'fan'. This indicates that the dynamics introduced by the grouping is effective, compelling the model to handle multiple consecutive occurrences of scenarios representing a single season.

[6]Note that this simple selection approach is applicable only to one-dimensional randomness (in our case, the wind-production capacity factor). With multidimensional randomness, or a need for more scenarios, a different approach would be needed. One popular choice is clustering methods such as k-medoids (Kaufman and Rousseeuw, 1990, Chapter 2) – a method similar to k-means, but using actual data points as cluster centres. For more methods see, for example, Bounitsis et al. (2022) or Kaut (2021).

- The 'mean+min_groups' variants can be more costly than 'full'. This could be because the scenario variants force the model to handle both the 'min' scenario and multiple occurrences of the 'mean' scenarios simultaneously (at the start of each season), while the actual sequence in the 'full' variant might be easier to manage.

The last point is important: Eq. (6) compels the model to consider multiple repetitions of the involved scenarios, up to the specified probability. As a result, the solutions have to handle many different permutations of the involved scenarios. The 'full' variant, on the other hand, allows the model to adapt the solution to the one scenario (historical data) included in the tree. Consequently, the scenario selection might yield more expensive solutions.

However, this does not necessarily mean that the scenarios lead to *better* solutions. For instance, consider a 12-week season represented by two weekly scenarios of equal weight, where the inventory level increases by 10 in the first scenario and decreases by 10 in the second. With a 5% limit on scenario repetitions, we get $\text{rep}_{s,sn,g} = 4$, so the model will have to handle up to four consecutive occurrences of either scenario. This will require the storage capacity to be at least 80, to allow the inventory to both increase and decrease by 40 units, assuming it starts at 40. If the two scenarios in reality tend to alternate and never occur more than twice consecutively, the actual storage-capacity requirement would be lower and so would be the total costs.

The right panels of Figs. 8 to 10 illustrate the sources of the costs. We can see that increased wind variability (i.e., more scenarios) is addressed by a combination of increased hydrogen storage and increased wind-production capacity. Specifically, in the '1×1' variant, the most expensive case is the one with largest storage, while in the '1×5' and '5×1' variants, it is the one with the largest wind park. This interplay between the two components makes it difficult to draw conclusions about the effects of the scenario structure on either of them alone.

5. Conclusions

This paper presents a methodology for modelling long-term storage within the multi-horizon modelling paradigm, including an approximation of the required storage capacities. This overcomes a significant barrier for the adoption of this modelling approach, thereby broadening its applicability to a wider range of problems.

The proposed formulation is implemented in the HyOpt optimization model, which is freely accessible from `https://gitlab.sintef.no/open-hyopt/`. The test case from Section 4 can be found at `https://gitlab.sintef.no/open-hyopt/test-case-2`. This includes scripts for solving the problems using pyHyOpt, a python interface to HyOpt, also available from the HyOpt page.

Acknowledgements

The research presented in this paper has been supported by the LowEmission project, funded by the Research Council of Norway under project number 296207. More information can be found at https://prosjektbanken.forskningsradet.no/project/FORISS/296207.

References

Hubert Abgottspon and Göran Andersson. Multi-horizon modeling in hydro power planning. *Energy Procedia*, 87:2–10, jan 2016. doi: 10.1016/j.egypro.2015.12.351.

Stian Backe, Mohammadreza Ahang, and Asgeir Tomasgard. Stable stochastic capacity expansion with variable renewables: Comparing moment matching and stratified scenario generation sampling. *Applied Energy*, 302:117538, November 2021. ISSN 0306-2619. doi: 10.1016/j.apenergy.2021.117538.

Stian Backe, Christian Skar, Pedro Crespo del Granado, Ozgu Turgut, and Asgeir Tomasgard. EMPIRE: An open-source model based on multi-horizon programming for energy transition analyses. *SoftwareX*, 17:100877, January 2022. ISSN 2352-7110. doi: 10.1016/j.softx.2021.100877.

Chiara Bordin and Asgeir Tomasgard. Smacs model, a stochastic multihorizon approach for charging sites management, operations, design, and expansion under limited capacity conditions. *Journal of Energy Storage*, 26:100824, December 2019. ISSN 2352-152X. doi: 10.1016/j.est.2019.100824.

Chiara Bordin, Sambeet Mishra, and Ivo Palu. A multihorizon approach for the reliability oriented network restructuring problem, considering learning effects, construction time, and cables maintenance costs. *Renewable Energy*, 168:878–895, May 2021. ISSN 0960-1481. doi: 10.1016/j.renene.2020.12.105.

Georgios L. Bounitsis, Lazaros G. Papageorgiou, and Vassilis M. Charitopoulos. Data-driven scenario generation for two-stage stochastic programming. *Chemical Engineering Research and Design*, 187:206–224, nov 2022. doi: 10.1016/j.cherd.2022.08.014.

Lars Hellemo, Kjetil Midthun, Asgeir Tomasgard, and Adrian Werner. Multi-stage stochastic programming for natural gas infrastructure design with a production perspective. In Horand I. Gassmann, Stein W. Wallace, and William T. Ziemba, editors, *Stochastic Programming: Applications in Finance, Energy, Planning and Logistics*, World Scientific Series in Finance, pages 259–288. World Scientific, 2013. doi: 10.1142/9789814407519_0010.

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., March 1990. doi: 10.1002/9780470316801.

Michal Kaut. Scenario generation by selection from historical data. *Computational Management Science*, 18(3):411–429, jun 2021. doi: 10.1007/s10287-021-00399-4.

Michal Kaut, Truls Flatberg, and Miguel M. Ortiz. The HyOpt model: Input data and mathematical formulation. techreport 2019:01439, SINTEF, 2019. URL https://hdl.handle.net/11250/2643389.

Richard Loulou and Antti Lettila. Stochastic programming and tradeoff analysis in TIMES. Technical report, IEA-ETSAP, 2016. URL https://iea-etsap.org/index.php/documentation.

Francesca Maggioni, Elisabetta Allevi, and Asgeir Tomasgard. Bounds in multi-horizon stochastic programs. *Annals of Operations Research*, 292(2):605–625, April 2019. ISSN 1572-9338. doi: 10.1007/s10479-019-03244-9.

Hans-Kristian Ringkjøb, Peter M. Haugan, Pernille Seljom, Arne Lind, Fabian Wagner, and Sennai Mesfun. Short-term solar and wind variability in long-term energy system models – a european case study. *Energy*, 209:118377, October 2020. ISSN 0360-5442. doi: 10.1016/j.energy.2020.118377.

Paula Rocha, Michal Kaut, and Afzal S. Siddiqui. Energy-efficient building retrofits: An assessment of regulatory proposals under uncertainty. *Energy*, 101:278–287, 2016. doi: 10.1016/j.energy.2016.01.037.

Pernille Seljom and Asgeir Tomasgard. Short-term uncertainty in long-term energy system models — A case study of wind power in Denmark. *Energy Economics*, 49: 157–167, 2015. doi: 10.1016/j.eneco.2015.02.004.

Christian Skar, Gerard Doorman, Gerardo Pérez-Valdés, and Asgeir Tomasgard. A multi-horizon stochastic programming model for the European power system. techreport 2/2016, CenSES, 2016. URL https://www.ntnu.no/censes/working-papers.

Lars Skaugen Strømholm and Raag August Sandal Rolfsen. Flexible hydrogen production : a comprehensive study on optimizing cost-efficient combinations of production and storage capacity to exploit electricity price fluctuations. mathesis, Norwegian School of Economics (NHH), 2021. URL https://hdl.handle.net/11250/2770501.

Hongyu Zhang, Asgeir Tomasgard, Brage Rugstad Knudsen, Harald G. Svendsen, Steffen J. Bakker, and Ignacio E. Grossmann. Modelling and analysis of offshore energy hubs. *Energy*, 261:125219, December 2022. ISSN 0360-5442. doi: 10.1016/j.energy.2022.125219.

Hongyu Zhang, Èric Mor Domènech, Ignacio E. Grossmann, and Asgeir Tomasgard. Decomposition methods for multi-horizon stochastic programming. in review, August 2023a.

Hongyu Zhang, Nicolò Mazzi, Ken McKinnon, Rodrigo Garcia Nava, and Asgeir Tomasgard. A stabilised benders decomposition with adaptive oracles for large-scale stochastic programming with short-term and long-term uncertainty. in review, 2023b.

Appendix

A. Mosel code for the storage modelling

Below, we present the HyOpt implementation of the presented storage-modelling approach, written in the FICO™ Mosel modelling language. This is a slightly cleaned version of the relevant section from file `HYOPT_model.mos`, available from the HyOpt model repository.

HyOpt implementation of storage modelling

```
! General storage modeling
declarations
  strat_end_level: array(ST_N) of linctr  ! end-of-strat-per storage
   level
end-declarations

declarations
  OSG_GroupWeight: array(STORAGES, ST_N, OSC_GROUPS) of real
  osc_group_level_change: array(STORAGES, ST_N, OSC_GROUPS) of linctr
  osc_group_end_level: array(STORAGES, ST_N, OSC_GROUPS) of mpvar
  osc_num_repeats: integer   ! number of repetitions
end-declarations

forall(s in STORAGES) do
  ! Storage level can not exceed total storage capacity (invested +
   existing)
  forall(sn in ST_N | exists(capacity(s,sn)), sc in SN_OP_SCENS(sn)) do
    InitStorageLevelCapConstr(s,sn,sc) :=
      init_storage_level(s,sn,sc) <= StorageUnitMult(s) *
   capacity(s,sn)
    forall (op in O_PERS(sc)) do
      Storage_capacity(s,sn,sc,op) :=
        storage_level(s,sn,sc,op) <= StorageUnitMult(s) *
   capacity(s,sn)
    end-do
  end-do

  ! Mass balance in storage
  forall(sn in ST_N, sc in SN_OP_SCENS(sn), op in O_PERS(sc)) do
    MassBalance_storage(s,sn,sc,op) :=
      storage_level(s,sn,sc,op) =
        if(op = FirstOpPer(sc), init_storage_level(s,sn,sc),
   storage_level(s,sn,sc,op-1)) +
        sum(n in NODES) FillEfficiency(s) *
   flow(n,s,StorageProduct(s),sn,sc,op) +
```

```
      - sum(n in NODES) 1/EmptyEfficiency(s) *
 flow(s,n,StorageProduct(s),sn,sc,op)
end-do

if not UseOpScenGroups then
  ! initial values and cyclical storage - depends on the time scope
  if StorageTimeScope(s) <> 'oper_scen' then
    forall (sn in ST_N) do
      strat_end_level(sn) := sum(sc in SN_OP_SCENS(sn))
 OperScenWeight(sn,sc) * (
       init_storage_level(s,sn,sc) + SPerOpMult(sn,sc) *
 (storage_level(s,sn,sc,LastOpPer(sc)) -
 init_storage_level(s,sn,sc)))
    end-do
  end-if
  case StorageTimeScope(s) of
    'oper_scen': do  ! storage tracking per oper. scenario
      ! control the initial storage
      if exists(InitialStorageFill(s)) then
        forall (sn in ST_N, sc in SN_OP_SCENS(sn)) do
          InitStorageLevelConstr(s,sn,sc) :=
            init_storage_level(s,sn,sc) = InitialStorageFill(s) *
 StorageUnitMult(s) * capacity(s,sn)
        end-do
      end-if
      ! cyclical storage
      if StorageIsCyclic(s) then
        forall(sn in ST_N, sc in SN_OP_SCENS(sn)) do
          InitStorageLevelConstr(s,sn,sc) :=
            init_storage_level(s,sn,sc) =
 storage_level(s,sn,sc,LastOpPer(sc))
        end-do
      end-if
    end-do

    'strat_node': do  ! storage tracking per node of the strategic
 tree
      ! control the initial storage
      if exists(InitialStorageFill(s)) then
        forall (sn in ST_N, sc in SN_OP_SCENS(sn)) do
          InitStorageLevelConstr(s,sn,sc) :=
            init_storage_level(s,sn,sc) = InitialStorageFill(s) *
 StorageUnitMult(s) * capacity(s,sn)
        end-do
      end-if
      ! cyclical storage
      if StorageIsCyclic(s) then
        forall(sn in ST_N) do
          forall(sc in SN_OP_SCENS(sn)) do
            InitStorageLevelConstr(s,sn,sc) :=
              init_storage_level(s,sn,sc) = strat_end_level(sn)
          end-do
```

```
        end-do
      end-if
    end-do

    'overall': do  ! storage tracking for the whole duration of the
 model
      ! tracking between strat. periods
      forall (sn in ST_N | ST_N_CH(sn).size > 0) do
        forall (snc in ST_N_CH(sn), sc in SN_OP_SCENS(snc)) do
          InitStorageLevelConstr(s,snc,sc) :=
            init_storage_level(s,snc,sc) = strat_end_level(sn)
        end-do
      end-do
      ! control the initial storage
      if exists(InitialStorageFill(s)) then
        forall (sn in {STRootN}, sc in SN_OP_SCENS(sn)) do
          InitStorageLevelConstr(s,sn,sc) :=
            init_storage_level(s,sn,sc) = InitialStorageFill(s) *
 StorageUnitMult(s) * capacity(s,sn)
        end-do
      end-if
      ! cyclical storage
      if StorageIsCyclic(s) then
        forall(sn in ST_N | ST_N_CH(sn).size = 0) do
          forall(sc in SN_OP_SCENS(STRootN)) do
            InitStorageLevelRootConstr(s,sn,sc) :=
              init_storage_level(s,STRootN,sc) = strat_end_level(sn)
          end-do
        end-do
      end-if
    end-do
  end-case

else
  ! UseOpScenGroups = true
  forall (sn in ST_N, osg in OSC_GROUPS | exists(OSC_GROUP_SCENS(sn,
 osg))) do
    OSG_GroupWeight(s,sn,osg) := sum(sc in OSC_GROUP_SCENS(sn, osg))
 OperScenWeight(sn,sc)
    create(osc_group_init_level(s,sn,osg))
    osc_group_level_change(s,sn,osg) := sum(sc in
 OSC_GROUP_SCENS(sn, osg)) OperScenWeight(sn,sc) /
 OSG_GroupWeight(s,sn,osg) * OpScenGroupDurH(sn,osg) /
 OperScenDurH(sc) * (storage_level(s,sn,sc,LastOpPer(sc)) -
 init_storage_level(s,sn,sc))
    OSG_GroupEndLevelDef(s,sn,osg) :=
      osc_group_end_level(s,sn,osg) = osc_group_init_level(s,sn,osg)
 + osc_group_level_change(s,sn,osg)

    ! handling scenario repetitions
    forall(sc in OSC_GROUP_SCENS(sn, osg)) do
      with p = OperScenWeight(sn,sc) / OSG_GroupWeight(s,sn,osg) do
```

```
        with maxRepeats = round(p * OpScenGroupDurH(sn,osg) /
OperScenDurH(sc)) do
          if p = 1 then
            osc_num_repeats := maxRepeats
          else
            osc_num_repeats := minlist(floor(ln(OSG_ScenRepeatProb)
/ ln(p)), maxRepeats)
          end-if
        end-do
      end-do
      if osc_num_repeats > 1 then
        forall(op in O_PERS(sc)) do
          ! storage levels in the last repetition must be within
bounds
          OSG_StorageLevelAdjMin(s,sn,sc,op) :=
            storage_level(s,sn,sc,op) + (osc_num_repeats - 1) *
(storage_level(s,sn,sc,LastOpPer(sc)) -
init_storage_level(s,sn,sc)) >=
            if(exists(StorageMinLevelRel(s)), StorageUnitMult(s) *
StorageMinLevelRel(s) * capacity(s,sn), 0)
          if exists(capacity(s,sn)) then
            ! upper bound only if we have capacity
            OSG_StorageLevelAdjMax(s,sn,sc,op) :=
              storage_level(s,sn,sc,op) + (osc_num_repeats - 1) *
(storage_level(s,sn,sc,LastOpPer(sc)) -
init_storage_level(s,sn,sc)) <=
              StorageUnitMult(s) * if(exists(StorageMaxLevelRel(s)),
StorageMaxLevelRel(s), 1) * capacity(s,sn)
          end-if
        end-do
      end-if
    end-do
  end-do

  ! inside each group, the initial storage level should be the same
  forall (sn in ST_N, osg in OSC_GROUPS, sc in OSC_GROUP_SCENS(sn,
osg)) do
    OSG_InitLevelInGroupConstr(s,sn,sc) :=
      init_storage_level(s,sn,sc) = osc_group_init_level(s,sn,osg)
  end-do

  ! storage-level tracking between groups - for all time scopes
  forall (sn in ST_N, osg in OSC_GROUPS | osg > FirstOpScGroup(sn)
and exists(OSC_GROUP_SCENS(sn, osg))) do
    OSG_StorageLevelSeqConstr(s,sn,osg) :=
      osc_group_init_level(s,sn,osg) = osc_group_end_level(s,sn,osg
- 1)
  end-do

  ! storage-level tracking between strategic periods/nodes
  if StorageTimeScope(s) = 'overall' then
    ! - init_level of next period is equal to the end_level from
```

```
  previous per.
     forall (sn in ST_N | exists(STPrevN(sn))) do
       OSG_OverallStorageLoop(s,sn) :=
         osc_group_init_level(s,sn,FirstOpScGroup(sn)) =
osc_group_end_level(s,STPrevN(sn),LastOpScGroup(STPrevN(sn)))
     end-do
   end-if

   if exists(InitialStorageFill(s)) then
     ! fixed initial storage fill
     case StorageTimeScope(s) of
       'oper_scen', 'osc_group': do
         ! storage tracking per oper. scenario or oper. scen. group
         forall (sn in ST_N, osg in OSC_GROUPS |
exists(OSC_GROUP_SCENS(sn, osg))) do
           OSG_InitStorageLevelGroup(s,sn,osg) :=
             osc_group_init_level(s,sn,osg) = InitialStorageFill(s) *
StorageUnitMult(s) * capacity(s,sn)
         end-do
       end-do

       'strat_node': do
         ! storage tracking per node of the strategic tree
         forall (sn in ST_N, osg in {FirstOpScGroup(sn)}) do
           OSG_InitStorageLevelGroup(s,sn,osg) :=
             osc_group_init_level(s,sn,osg) = InitialStorageFill(s) *
StorageUnitMult(s) * capacity(s,sn)
         end-do
       end-do

       'overall': do
         ! storage tracking for the whole duration of the model
         forall (sn in {STRootN}, osg in {FirstOpScGroup(sn)}) do
           OSG_InitStorageLevelGroup(s,sn,osg) :=
             osc_group_init_level(s,sn,osg) = InitialStorageFill(s) *
StorageUnitMult(s) * capacity(s,sn)
         end-do
       end-do
     end-case
   end-if  ! exists(InitialStorageFill(s))

   if StorageIsCyclic(s) then
     case StorageTimeScope(s) of
       'oper_scen': do
         ! storage level loops in every oper. scenario
         forall(sn in ST_N, sc in SN_OP_SCENS(sn)) do
           InitStorageLevelConstr(s,sn,sc) :=
             init_storage_level(s,sn,sc) =
storage_level(s,sn,sc,LastOpPer(sc))
         end-do
       end-do
```

```
      'osc_group': do
        ! storage level loops in each oper. scenario group
        forall (sn in ST_N, osg in OSC_GROUPS |
  exists(OSC_GROUP_SCENS(sn, osg))) do
          osc_group_level_change(s, sn, osg) = 0
        end-do
      end-do

      'strat_node': do
        ! storage level loops in each strategic node
        forall(sn in ST_N) do
          OSG_InitStorageLevelStN(s,sn) :=
            osc_group_init_level(s, sn, FirstOpScGroup(sn)) =
  osc_group_end_level(s, sn, LastOpScGroup(sn))
        end-do
      end-do

      'overall': do
        ! tracking between strat. periods
        forall(sn in ST_N | ST_N_CH(sn).size = 0) do
          OSG_InitStorageLevelRoot(s) :=
            osc_group_init_level(s, STRootN, FirstOpScGroup(sn)) =
  osc_group_end_level(s, sn, LastOpScGroup(sn))
        end-do
      end-do
    end-case
  end-if
end-if  ! UseOpScenGroups

! Minimum storage level .. abs
if exists(NodeParamValue(s, 'Min level abs')) then
  forall(sn in ST_N, sc in SN_OP_SCENS(sn), op in O_PERS(sc)) do
    Storage_min_level_abs(s,sn,sc,op) :=
      storage_level(s,sn,sc,op) >= StorageUnitMult(s) *
  NodeParamValue(s, 'Min level abs')
  end-do
end-if

! Minimum storage level .. relative
if exists(StorageMinLevelRel(s)) then
  forall(sn in ST_N, sc in SN_OP_SCENS(sn), op in O_PERS(sc)) do
    Storage_min_level_rel(s,sn,sc,op) :=
      storage_level(s,sn,sc,op) >= StorageUnitMult(s) *
  StorageMinLevelRel(s) * capacity(s,sn)
  end-do
end-if

! Maximum storage level .. relative
if exists(StorageMaxLevelRel(s)) then
  forall(sn in ST_N, sc in SN_OP_SCENS(sn), op in O_PERS(sc)) do
    Storage_max_level_rel(s,sn,sc,op) :=
      storage_level(s,sn,sc,op) <= StorageUnitMult(s) *
```

```
    StorageMaxLevelRel(s) * capacity(s,sn)
     end-do
  end-if
end-do
```