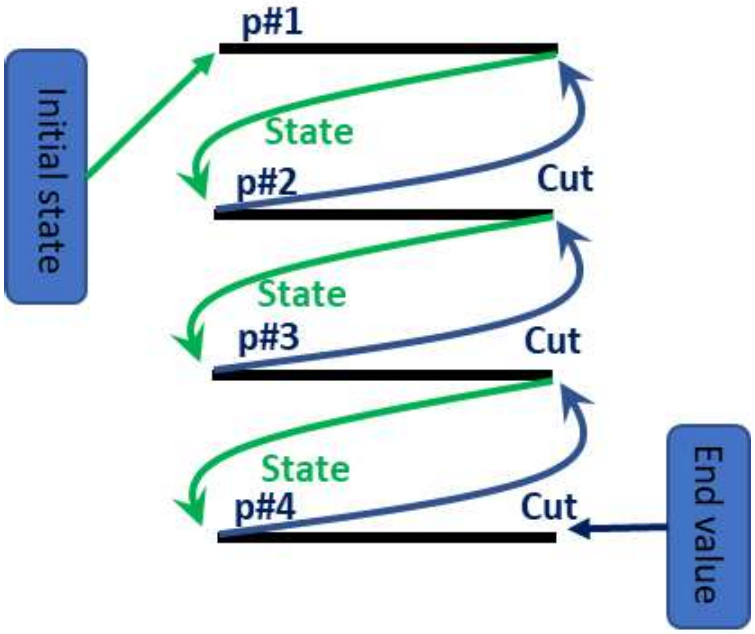

Report

Relaxation Techniques and Temporal Decomposition of Scenarios in Fundamental Power Market Models

Author(s):
Arild Helseth

Report No:
2022:00960 - Unrestricted
Client(s) (pos partner):
RAKETT project

SINTEF Energy Research
Postal address:
Postboks 4761 Torgarden
7465 Trondheim
Switchboard: +47 45456000
energy.research@sintef.no

Enterprise /VAT No:
NO 939 350 675 MVA

Report

Relaxation Techniques and Temporal Decomposition of Scenarios in Fundamental Power Market Models

KEYWORDS
Power market
Hydropower scheduling
Linear programming
Decomposition

VERSION
1.0

DATE
2022-10-28

AUTHOR(S)
Arild Helseth

CLIENT(S)
RAKETT project

CLIENT'S REFERENCE
Klikk eller trykk her for å skrive inn tekst.

PROJECT NO.
502002634

NO. OF PAGES/APPENDICES
35

SUMMARY
This report elaborates on the possibilities for reducing computation time in the Fansi power market model by applying relaxation techniques and temporal decomposition. Findings and conclusions are supported by computational experiments conducted on a realistic large-scale dataset.

Relaxation of reservoir balance constraints are investigated, and an effective heuristic method is presented.

Both serial and parallel implementations of Benders decomposition applied to a deterministic linear programming problem are investigated. Although the parallel implementation shows potential for improving computational performance, further investigations are recommended.

PREPARED BY
Arild Helseth

SIGNATURE
Arild Helseth
Arild Helseth (Nov 1, 2022 09:59 GMT+1)

CHECKED BY
Birger Mo

SIGNATURE
bm
bm (Nov 1, 2022 10:13 GMT+1)

APPROVED BY
Knut Samdal

SIGNATURE
Knut Samdal
Knut Samdal (Nov 8, 2022 10:45 GMT+1)

REPORT NO.
2022:00960

ISBN
978-82-14-07591-5

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
1.0	2022-11-01	First version

Table of contents

1 Introduction

The primary goal of the RAKETT project (2020-2022) is to investigate decomposition techniques suited to reduce the computation time in the fundamental power market model Fansi [1]. The power market model is designed for hydro-dominated systems, solving the so-called *hydrothermal scheduling problem*. Fansi builds and solves two-stage stochastic optimization problems repeatedly, arranged in a simulator. Each two-stage problem is formulated as a linear programming (LP) problem that we refer to as a scenario fan problem (SFP). The SFP is decomposed in a first-stage master problem and several second-stage scenario problems, and coordinated by using Benders decomposition, more specifically the L-shaped method described in [2]. Depending on the time discretization and functionality represented, both the master and scenario problems can become large-scale LP problems with millions of variables and constraints. While the master problem covers a time horizon of one week, the scenario problem horizons may range from a season to several years ahead.

In this report we are concerned with *temporal* decomposition of *scenarios*. By temporal decomposition we mean decomposition along the time axis, and by scenarios we broadly refer to a deterministic LP problem describing the hydrothermal scheduling problem over a sequence of time periods. According to the definition of scenarios above, both the master and scenario problems solved in Fansi can be characterized as scenarios. As a prerequisite for this work, the Fansi SFP is already decomposed in time through a master and multiple scenario problems. This work studies the further decomposition of the master and scenario problems. Such extended decomposition could naturally be a part of the existing decomposition of the SFP.

1.1 Related Work

Techniques for spatial decomposition has been explored within the RAKETT project. The main challenge of applying spatial decomposition to the hydrothermal scheduling problem is to efficiently optimize hydropower watercourses or regions separately.

In [3] Lagrangian Relaxation was applied to the hydrothermal scheduling problem following the algorithm described in [4]. A different form of spatial decomposition was presented in [5] using Benders decomposition principle. The technique outlined in [5] was later applied within the stochastic dual dynamic programming (SDDP) algorithm in [6].

1.2 Report Structure

This report is organized in two main parts. First, we study the potential for reducing computation time by relaxing the reservoir balance constraints. These constraints are typically included to keep track of the state (reservoir volumes) over time. With fine time resolution and for large reservoir storages, such constraints are often not binding and can be omitted or *relaxed* (i.e. added if needed) to reduce computation time. Although not decomposition per se, extensive testing of the impact of constraint relaxation fits well within the topic of temporal decomposition and can be an integral part of such decomposition schemes. In the second part, we study temporal decomposition for both short (one week) and long (one year) scenarios, organizing the algorithms in both serial and parallel mode. Computational experiments are reported throughout the report, using the hard- and software defined in Section 1.3 and applied to the data described in Section 1.4.

1.3 Computational Framework

All computational experiments are conducted using the programming language Julia (v 1.5.3) and the mathematical modelling language JuMP (v 0.21.5) together with the optimization solver CPLEX (v 12.10). Julia/JuMP facilitates flexible model building and experimentation and allows updating LP problems in memory. Thus, it is well suited for the type of tests conducted here. The dual Simplex algorithm was generally found to be the fastest algorithm for solving the type of problems defined here and is therefore used in all computational experiments. Presolve functionality is activated in all tests and "warm starting" the solution processes using a previous basic solution is applied when possible.

All tests were conducted on a laptop PC with 64 GB RAM and an Intel Core i7-9850H processor with maximal frequency of 4.60 Hz (turbo) and 6 cores, each with 2 threads. The PC was not loaded with additional activities during testing.

1.4 Data

SINTEFs "Hydrocen 2030 Low Emission" dataset was applied. It represents a scenario for 2030 stadium of the power system. The dataset covers the Northern-European power system with 57 price areas, bounded by Great Britain in west, France in south, and Poland in East).

- 1093 hydropower modules, many of which has PQ-curves with multiple efficiency segments.
- No exogenous price areas.

Variants of this dataset is actively used and improved within ongoing research projects at SINTEF. The dataset is further described in [7].

2 Relaxation of Reservoir Balances

We define a scenario as a least cost dispatch problem covering multiple time periods. The problem is formulated as an LP problem, initially with the following basic set of constraints:

Objective: Minimize costs
Constraints:

- Reservoir balances for all hydropower modules and time steps
- Power balances for all price areas and time steps
- End-of-horizon valuation by Benders cuts

Reservoir balances provide time couplings due to reservoir storage. Additional constraints can be added to further limit the operation of the system. Examples of such constraints are:

- Ramping on discharge
- Ramping on cables
- Time delay in rivers
- Start/stop thermal units

2.1 Hydropower System Representation

For representations of the Nordic system, variables and constraints associated with the detailed hydropower system will contribute substantially to the LP size. Typically, such representations will comprise more than 1000 hydropower modules with attributes indicated in Figure 1.

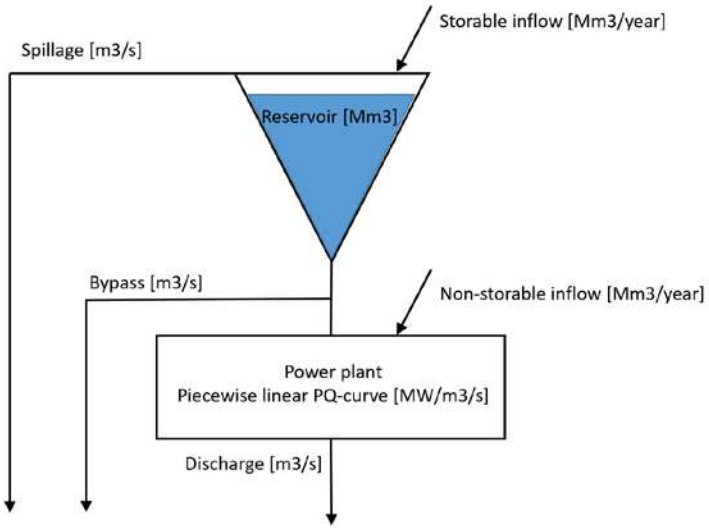

Figure 1 A hydropower module.

Storable and non-storable inflows are input parameters. Water flows can be classified as spillage (q_k^S), bypass (q_k^B), and discharge (q_k^D). Variables are associated with all water flows. Power is generated in the plant (or station), represented by a piecewise linear discharge-to-power (PQ) curve with *NS* segments. The

discharge flow is replaced by variables per segment $\sum_{s=1}^{NS} q_{sk}^{D}$ to represent the PQ-curve. Hydropower generation for a time step *k* is represented as:

$$p_k = \sum_{s=1}^{NS} \eta_s q_{sk}^{D} \tag{1.1}$$

Where η_s are the efficiencies (MW/m3/s) per discharge segment.

Normally the water volume is represented by a variable (v_k) that is computed by use of a reservoir balance. Water enters the reservoir from upstream modules $u \in \Omega$ or regulated inflow I_k and leaves the reservoir through discharge, bypass and spillage.

$$v_k - v_{k-1} + \sum_{s=1}^{NS} q_{sk}^{D} + q_k^{B} + q_k^{S} - \sum_{u \in \Omega} \left(\sum_{s=1}^{NS} q_{suk}^{D} + q_{uk}^{B} + q_{uk}^{S} \right) = I_k \tag{1.2}$$

The equation (1.2) is slightly adjusted and supplement with an additional constraint (1.4) if the hydropower module has non-storable inflow. This is done to ensure separation between storable and non-storable inflow.

$$v_k - v_{k-1} + q_k^{R} + q_k^{S} - \sum_{u \in \Omega} \left(\sum_{s=1}^{NS} q_{suk}^{D} + q_{uk}^{B} + q_{uk}^{S} \right) = I_k \tag{1.3}$$

We introduce a release variable q_k^{R} that further enters the balance:

$$q_k^{B} + \sum_{s=1}^{NS} q_{sk}^{D} - q_k^{R} = U_k \tag{1.4}$$

The reservoir balance in (1.2) and non-storable inflow balance in (1.4) has so far in Fansi been defined for each time step *k*. While it makes sense that (1.4) should be controlled per time step, the balance equations in (1.2) can often be relaxed, since reservoir volumes are well within their boundaries within the defined scenario length.

For a system with 1000 hydropower modules and 168 time steps, 168.000 reservoir constraints are needed. Considering a PQ-curve with 3 segments, each module needs 6 variables, leading to 1.008.000 variables. Thus, the representation of the hydropower system alone becomes an LP problem of substantial size.

2.2 Problem Simplification

We will now discuss the potential for reducing the number of reservoir variables and balances.

2.2.1 Reducing the Number of Water Flow Variables

The fine time discretization is important to capture the price formation and is therefore also important for models aimed at water value computation. Prices are obtained as dual values from the power balances,

and we know that hydropower generation from (1.1) contributes to the power balances. Thus, it is important to allow discharge per time step as decision variables.

In this work we will formulate all water release decisions per time step. However, there is potential for aggregating variables representing the bypass and spillage waterways. To do so one needs to carefully consider:

- Representing decisions as water volumes (Mm3/aggregated step) rather than a flow (m3/s)
- Matching the resolution of inflow, which is typically a mix of daily and weekly.

2.2.2 Reducing the Number of Reservoir Variables and Constraints

The reservoir volume variable is primarily needed for two reasons; to control if the volume is within the allowed boundaries, and to define the water left in the storage at the end of the planning horizon. Reservoir volumes may be subject to time-dependent boundaries with weekly granularity. For the largest reservoirs there is often no need to check the intra-week storage volume since they are not close to their boundaries. Thus, we suggest formulating reservoir balances per week *t* rather than per time step. Together with bypass and spillage decisions per week, the reservoir balance in (1.2) reduces to:

$$v_t - v_{t-1} + \sum_{k=1}^{NK}\left(\sum_{s=1}^{NS} q_{sk}^{D} + q_k^{B} + q_k^{S}\right) - \sum_{u\in\Omega}\sum_{k=1}^{NK}\left(\sum_{s=1}^{NS} q_{suk}^{D} + q_{uk}^{B} + q_{uk}^{S}\right) = \sum_{k=1}^{NK} I_k \qquad (1.5)$$

With this simplification, the number of constraints in our above example is reduced to from 168.000 to 1.000 and the number of variables from 1.008.000 to 841.000.

Applying weekly water balances to all reservoirs would obviously overestimate the flexibility of several reservoirs. The overestimation depends on several factors such as:

- Maximum reservoir capacity
- Maximum discharge capacity
- Initial reservoir level
- Inflow in the planning period
- Topology, including upstream releases and downstream constraints

From the list above, one can observe that the need for finer time resolution depends both on the physical characteristics of the hydropower module in questions, its location in the watercourse, and the state of the system (initial volume, inflow).

In the tests performed here we chose to include both weekly $v_t, \forall t \in T$ and intra-weekly $v_t, \forall k \in K$ variables. We focus on the reduction of reservoir constraints and not variables.

2.3 Relaxation Principle

The simplification related to the weekly reservoir balances can be controlled by relaxation techniques. At any time step k* the reservoir volume can be computed as:

$$v_{k*} = v_0 + \sum_{k=1}^{K} I_k + \sum_{u\in\Omega}\sum_{k=1}^{K}\left(\sum_{s=1}^{NS} q_{suk}^{D} + q_{uk}^{B} + q_{uk}^{S}\right) - \sum_{k=1}^{K}\left(\sum_{s=1}^{NS} q_{sk}^{D} + q_{k}^{B} + q_{k}^{S}\right) \quad (1.6)$$

Where all values on the right-hand side are parameters or decisions made in a first LP solution. The reservoir volume can then be checked according to its boundaries $\left[\underline{V}_t, \overline{V}_t\right]$.

Constraints of type (1.9) can be added to the LP problem if the lower (1.7) or upper (1.8) boundaries are violated for a time step k*.

$$v_{k*} \geq \underline{V}_t \quad (1.7)$$

$$v_{k*} \leq \overline{V}_t \quad (1.8)$$

$$\underline{V}_t - v_0 - \sum_{k=1}^{k^*} I_k \leq \sum_{u\in\Omega}\sum_{k=1}^{k^*}\left(\sum_{s=1}^{NS} q_{suk}^{D} + q_{uk}^{B} + q_{uk}^{S}\right) - \sum_{k=1}^{k^*}\left(\sum_{s=1}^{NS} q_{sk}^{D} + q_{k}^{B} + q_{k}^{S}\right) \leq \overline{V}_t - v_0 - \sum_{k=1}^{k^*} I_k \quad (1.9)$$

The inequality constraint in (1.9) can be added for each module and each time step with violated reservoir boundaries. This process is not likely computationally efficient for the following reasons:

1) The number of variables in (1.9) increases with k*, leading to dense constraints.
2) Several constraints with almost identical variables will be formulated.
3) If a constraint is added for k*, chances are that the constraint violation will occur in a nearby timestep in the next iteration.

2.4 Proposed Relaxation Scheme

Rather than adding constraints for each detected violation, we developed a simpler scheme based on two iterations. The algorithmic steps are described below:

1) Define a threshold VThres
2) Solve the relaxed LP where all reservoir balances for modules with Vmax <= VThres have reservoir balances per time step while the remaining ones have balances per week.
3) From the solution, check intraweek reservoir balances for reservoirs with Vmax > VThres according to (1.9).
4) For reservoirs with any intraweek reservoir limit violations: Add intraweek constraints for all time steps.
5) Resolve LP problem.

2.5 Computational Experiments

The relaxation scheme described in Section 2.4 was verified on the "Hydrocen 2030 Low Emission" dataset briefly described in Section 1.4.

Seven scenario cases were defined as described in Table 1, starting in weeks 9, 31 and 45 with initial reservoir levels obtained for these weeks from a previous run with the Fansi model. The cases vary in length (number of weeks), in time resolution (hours per time step), and in weather year. The year 1969 was dry while 2011 was wet, and 2013 in between.

Table 1 Scenario cases.

Case no	Number of weeks	Time resol. [h]	Start week	Weather year
1	10	6	9	1969
2	10	6	31	2011
3	8	3	9	1969
4	8	3	31	2011
5	1	1	9	1969
6	1	1	31	2011
7	1	1	45	2013

Note that when applying the iterative approach described in Section 2.4, the second iteration benefits from warm start when solving the LP problems.

2.5.1 System Costs

First, we study the optimal objective value (minimum system costs) after both iterations for different values of VThres, as shown in Figure 2 and Figure 3. The blue points show the objective values achieved in the first iterations, while the orange points show objective values from the second iteration. When VThres is higher than the largest maximum reservoir volume, there is no need for a second iterations because all constraints are added in the first iteration. As expected, the difference in objective value between the first and second iteration is highest for low values of VThres.

Note that the dataset contains some few hydropower modules with large (aggregated per country) reservoirs and with discharge capacities so high that these reservoirs can be emptied within a few weeks. Some of these modules has reservoirs exceeding 1000 Mm3 storage capacity. The presence of these modules necessitates either setting VThres high or conducting the second iteration so that intra-week reservoir constraints are obeyed.

While cases 1-4 spans several weeks, cases 5-7 span one week only. The minimum system costs after the first and second iteration for cases 5, 6, and 7 are shown in Figure 4 and Figure 5. For cases 5 and 7 we found that the by selecting VThres >= 200 Mm3, there seems to be no need for an iterative approach with respect to finding the optimal solution. This is different in Case 6 in where further iterations at low VThres values could be beneficial.

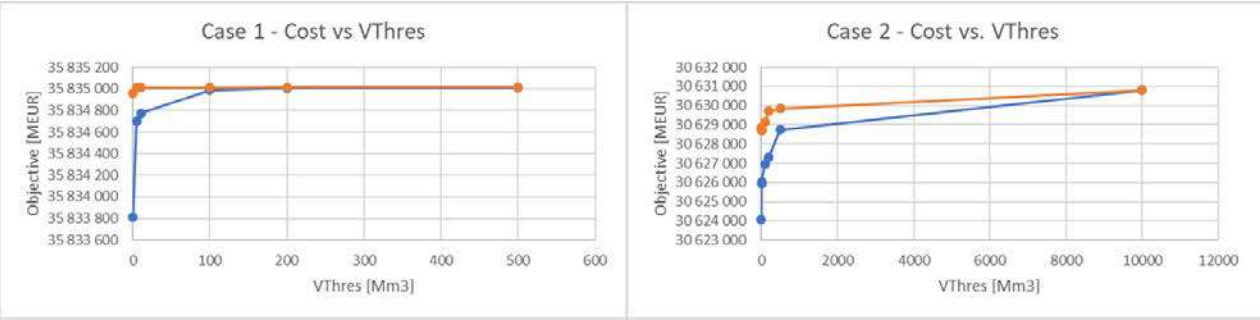

Figure 2 Objective function for different values of VThres for Cases 1 and 2. First iteration in blue and second in orange.

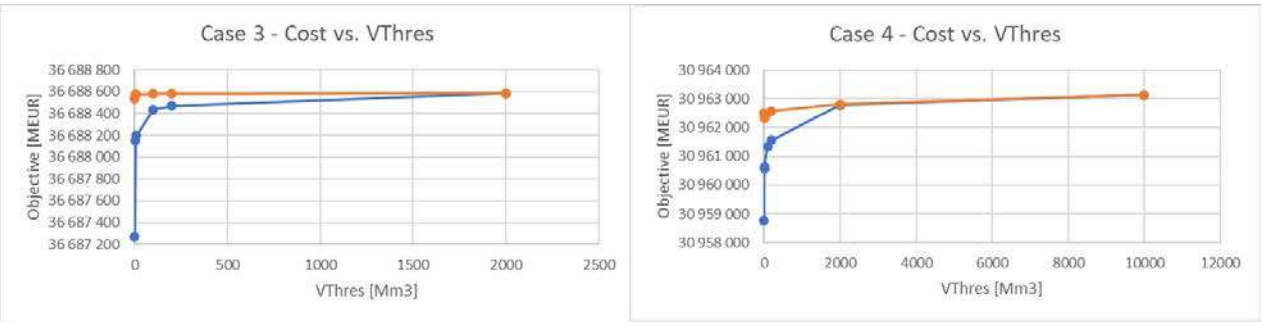

Figure 3 Objective function for different values of VThres for Cases 3 and 4. First iteration in blue and second in orange.

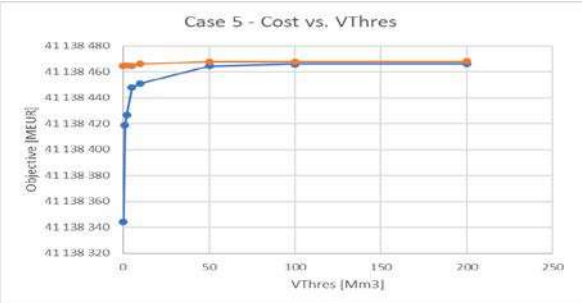

Figure 4 Objective function for different values of VThres for Case 5. First iteration in blue and second in orange.

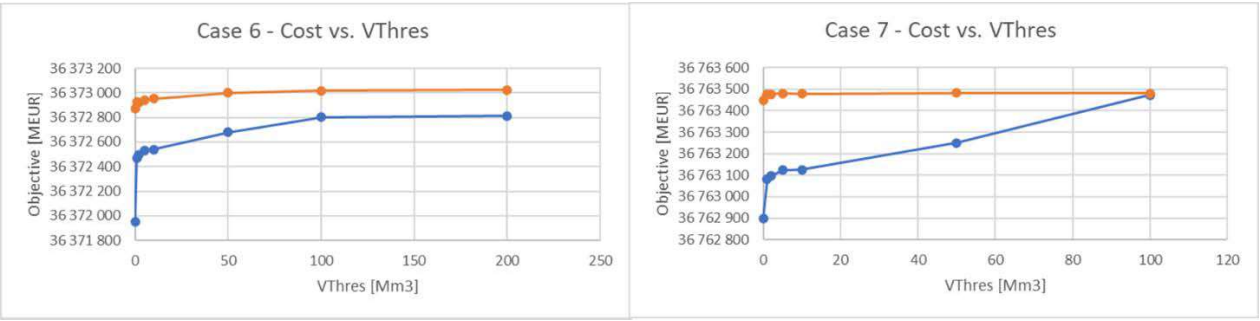

Figure 5 Objective function for different values of VThres for Cases 6 and 7. First iteration in blue and second in orange.

2.5.2 Computational Time

Figures 6, 7 and 8 show the time spent by CPLEX solving the optimization problems in the two iterations for cases 1-2, 3-4, and 5-7, respectively. From these figures we observe a substantial potential for reducing computation time by selecting a low value on VThres. This potential is substantial both for the weekly scenarios (5-7) and the ones with longer horizon (1-4). Note that the higher VThres is, the less work is needed when resolving LP problem in the second iteration, since few extra constraints are added. Thus, when VThres=10000 Mm3, all time is spent solving the first iteration.

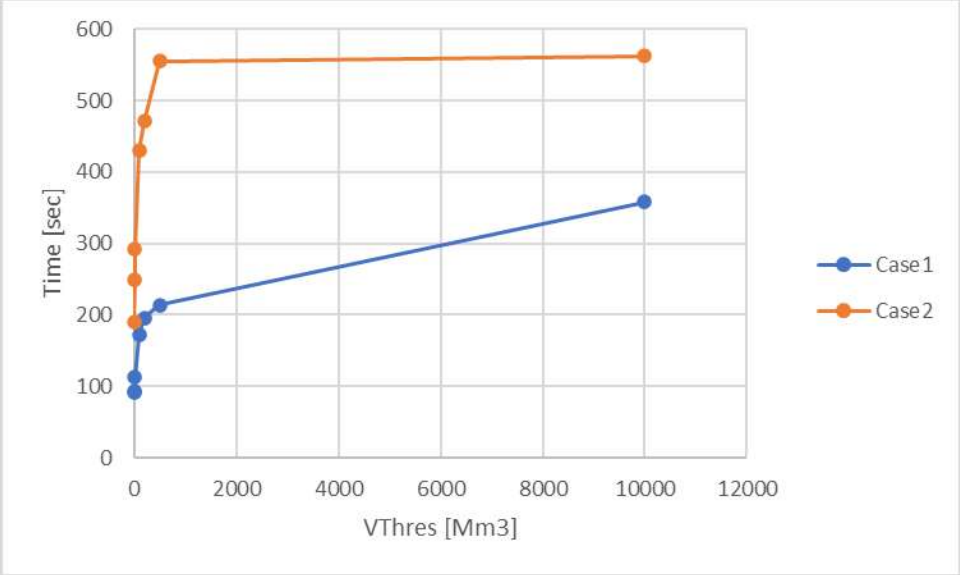

Figure 6 Total time spent by solver for different values of VThres for cases 1 and 2.

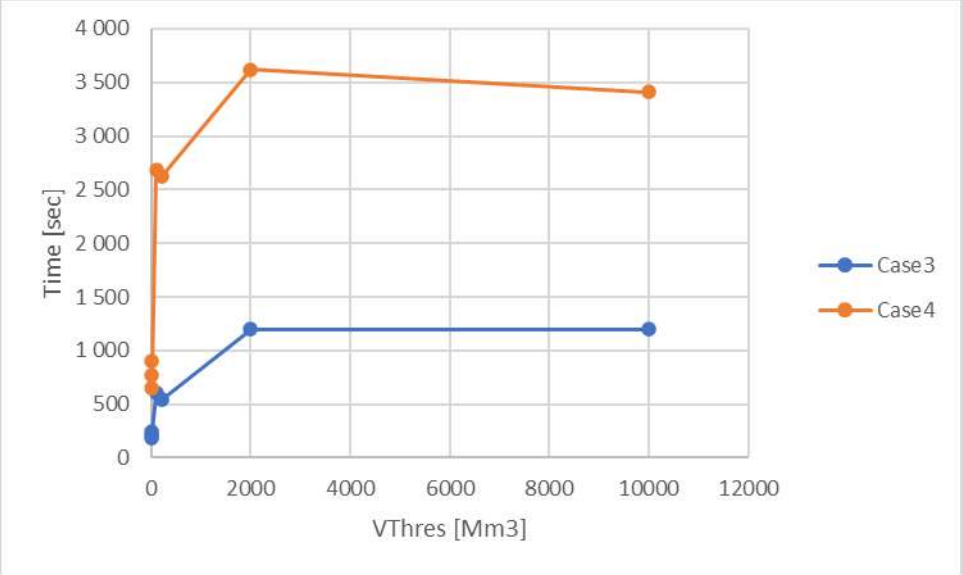

Figure 7 Total time spent by solver for different values of VThres for cases 3 and 4.

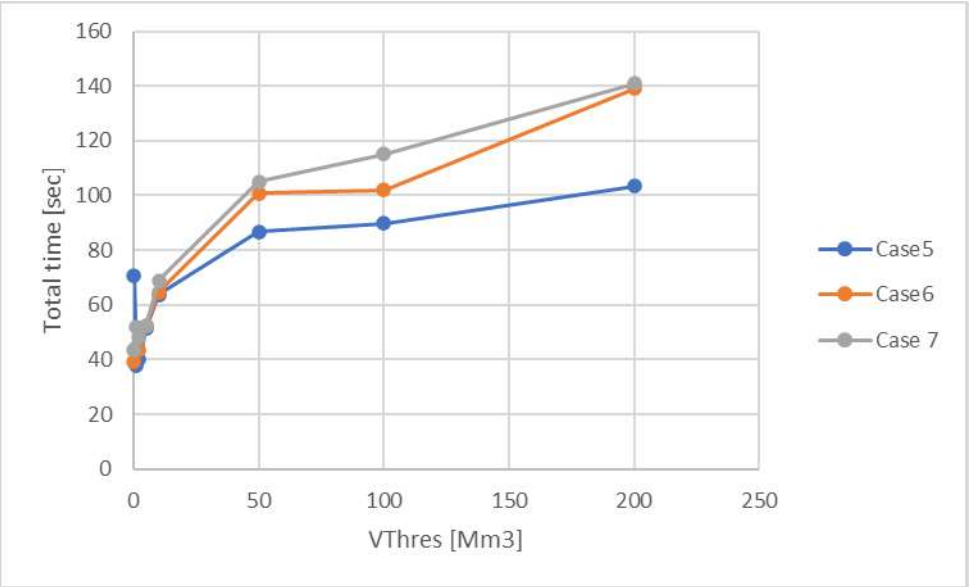

Figure 8 Total time spent by solver for different values of VThres for cases 5-7.

The time spent by CPLEX per iteration is broken down per iteration in Figure 9 and Figure 10.
The sharp increase in computation time with increasing VThres (at low VThres values) in the first iteration is noteworthy.

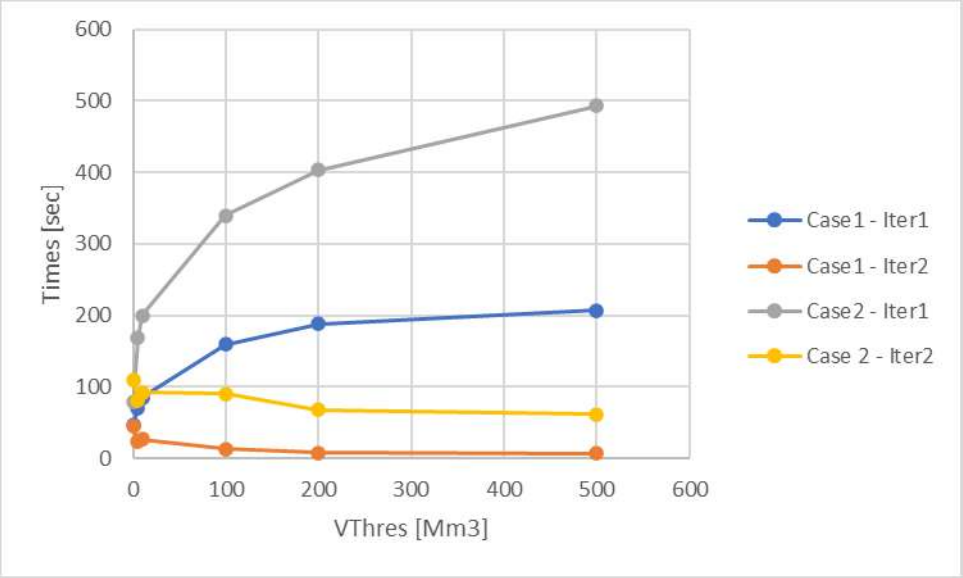

Figure 9 Time spent by solver per iteration for cases 1 and 2.

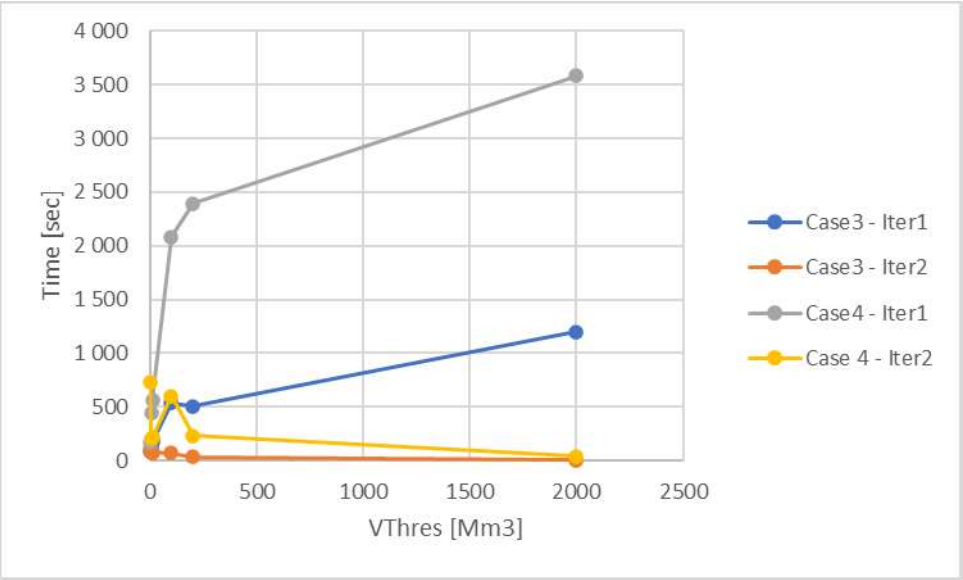

Figure 10 Time spent by solver per iteration for cases 3 and 4.

2.5.3 Prices

Figure 11 and Figure 12 show the simulated prices per time step for the 3 first weeks for area NORGEMIDT in case 3 and 4, respectively. Prices obtained after the first and second iteration with VThres=0 is compared with the "full" solution, i.e., where all reservoir balances are included. Area 8 was one of the areas with largest differences, but still differences are modest. While results from iteration 1 deviates some places, iter 2 closes the gap for most time steps.

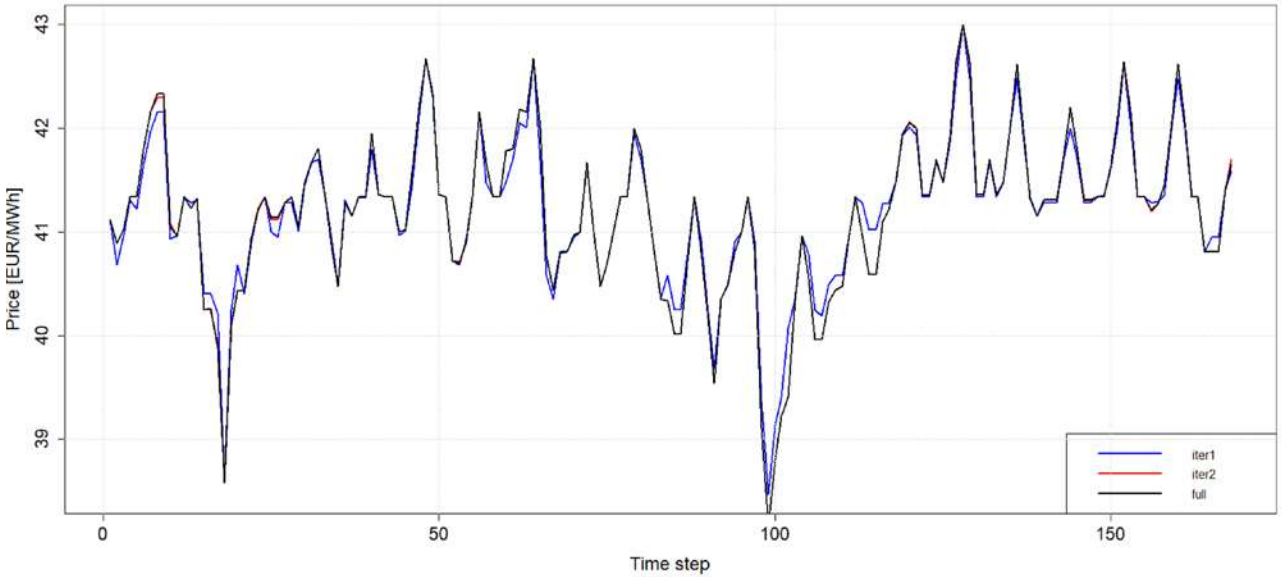

Figure 11 Simulated power prices in first and second iteration for NORGEMIDT in case 3 (week 9) with VThres=0.

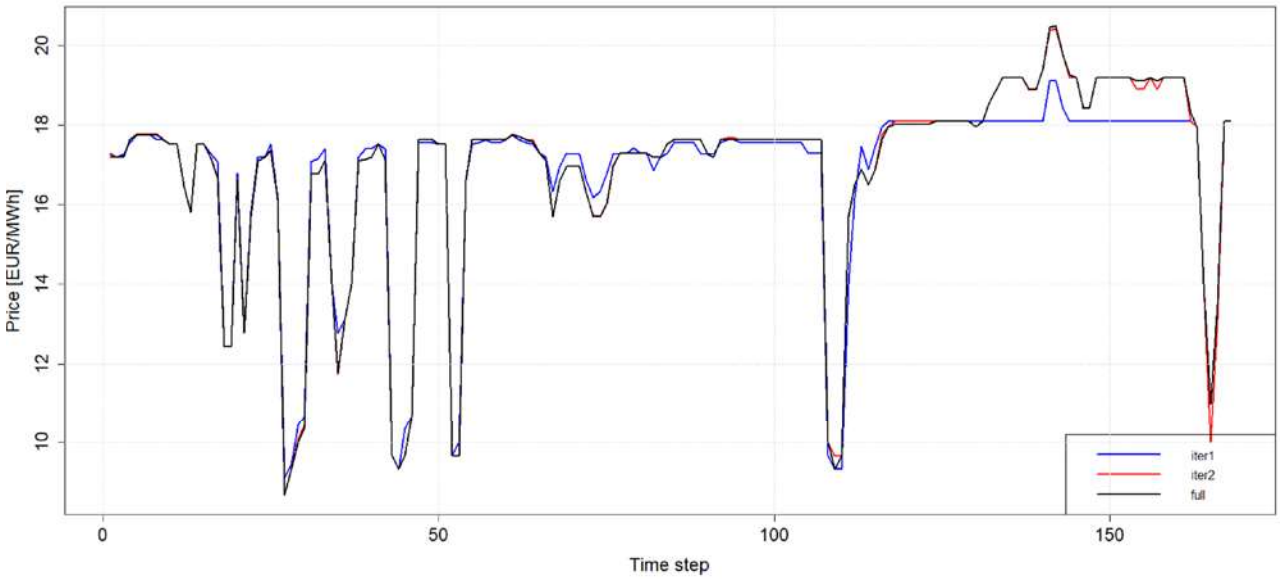

Figure 12 Simulated power prices in first and second iteration for NORGEMIDT in case 4 (week 31) with VThres=0.

2.5.4 Summary

We have presented computational times and results in terms of objective value (minimum cost) and power prices resulting from the relaxation algorithm presented in Section 2.4. The lowest computation times are in general found with Vthres=0. However, the quality of the results, as shown by the minimum cost plots in Figure 2-Figure 5 indicates that with very low values of VThres, all the systems costs are underestimated after the second iteration. Thus, a small positive value of VThres provides the best trade-off between computation time and quality of results.

3 Temporal Decomposition – Serial Benders

3.1 Introduction

In the context of Fansi, temporal decomposition is already conducted when solving the SFP as a two-stage stochastic LP, decoupling the master and scenario problems. Further temporal decomposition can be facilitated as part of the master and/or scenario problems, as illustrated in Figure 13. The stapled red lines indicate further splitting the master problem into subproblems, while stapled black lines indicates decomposition of the scenario problem.

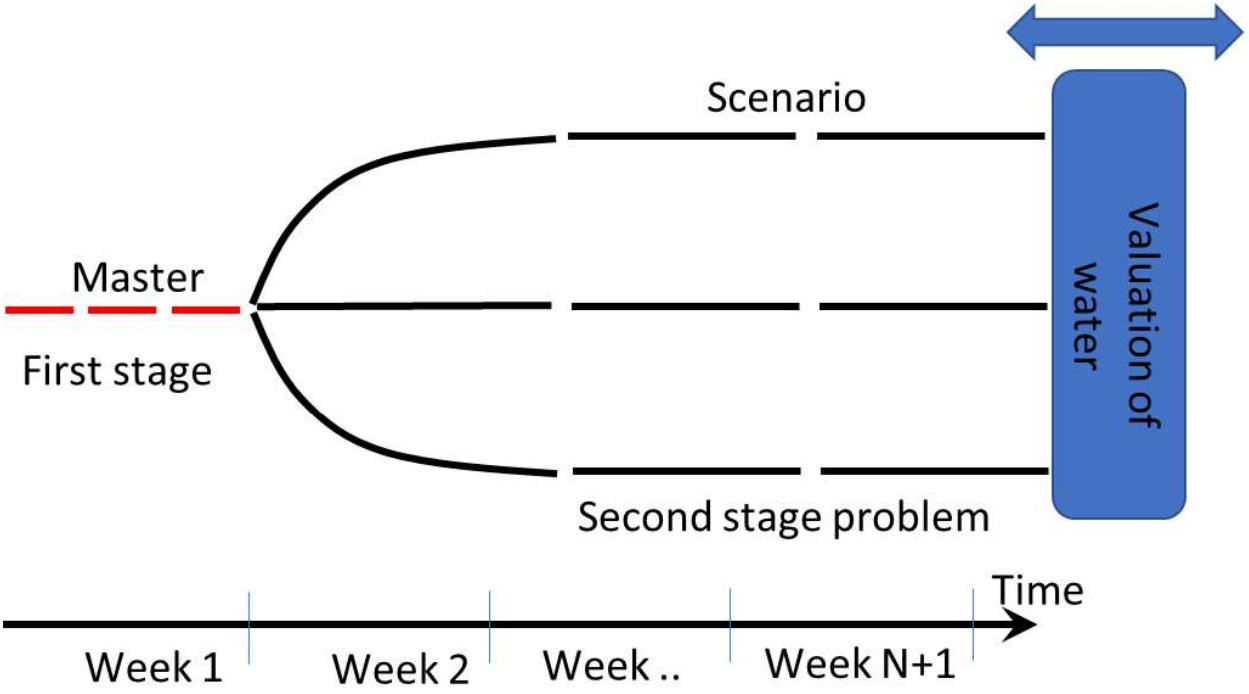

Figure 13 Further temporal decomposition of the SFP.

Both the master and scenario problems solved in the SFP decomposition conducted in Fansi are 'scenarios' in the context used here[1]. Thus, when studying the temporal decomposition of a scenario in the following, the findings may have application to both the master and scenario problems.

A standard multi-stage Benders decomposition algorithm [2] can be applied to decompose the scenario, as illustrated in Figure 14. Through iterations, system states are passed forward and Benders cuts are passed backwards.

Although the decomposition algorithm is sequential in nature, previous research has shown that parallel processing can be applied to speed up computation time [8]. This parallel decomposition scheme will be explored in Section 4.

[1] A deterministic LP problem describing the hydrothermal scheduling problem over a sequence of time periods.

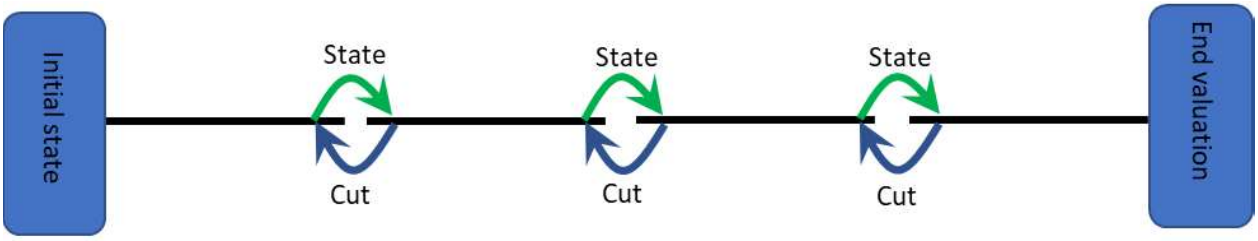

Figure 14 Temporal decomposition illustrated.

For all tests presented in the following we assume that an end-valuation in the form of Benders cuts exists and can be used to valuate the end-of-horizon storage.

In the research project PRIBAS, a simplified decomposition algorithm was applied to speed up the computation time for solving a weekly optimization problem. Each stage (day) was optimized separately interpolating in the end-valuation (Benders cuts) available for the end and beginning of the week. This process is described in detail in [9]. In this work we will evaluate the simplified version of this algorithm illustrated in Figure 15. The stage problems are simulated in a single forward simulation, each seeing the end-valuation. Note that we do not interpolate in cost functions as was done in PRIBAS. We refer to this approach as a *PRIBAS-iteration*[2].

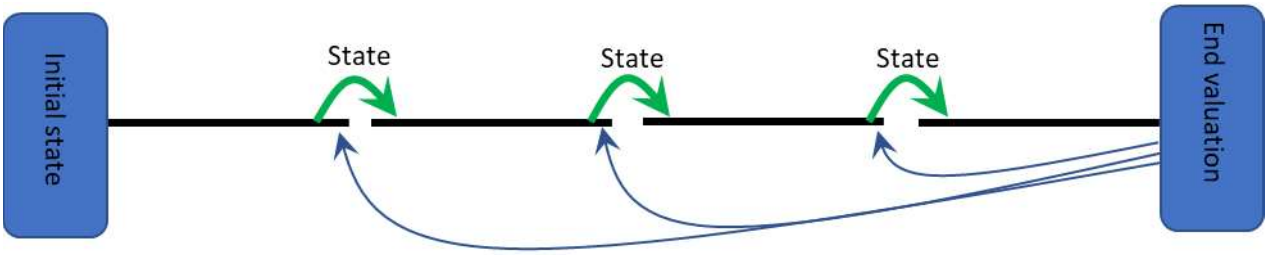

Figure 15 Illustration of a PRIBAS iteration.

3.2 Algorithm Description

In the following we describe the serial Benders decomposition algorithm. As a starting point we decide how many stages (NSub) each scenario should be divided into. As an example, the scenario in Figure 14 is divided in 4 stages.

In serial mode, the decomposition of a scenario is conducted as forward and backward iterations:

1) **Forward:** Starting at the initial state, one simulates forward in time stage-by-stage. The final state at a stage (reservoir volume and others) is passed forward to the next stage.
2) **Backward:** Starting at the end of the scenario and using the simulated state trajectory from step 1 one iterates backwards in time. Benders cuts are created in each stage (except from the first one) and passed backwards in time.

Convergence can be checked in the forward iteration: If the minimum cost seen from the first stage equals the cost found for all stages, convergence can be declared.

[2] Note that PRIBAS interpolated in cost functions (Benders cuts) available at the beginning and end of the week. Here, we only use cuts at the end of the week without interpolation.

We dedicate one process per stage problem to ensure that the problem is read once and that warm starts can be used. The two steps are executed in a sequence, as illustrated in Figure 16. In the forward iteration the processes are invoked in the sequence p#1→p#2→p#3→p#4, while p#4→p#3→p#2→p#1 are invoked in the backward iteration.

To avoid myopic solutions of stage problems in the first iterations when no Benders cuts have been created, all first iterations are carried out as PRIBAS-iterations, i.e., with the existing end valuation (Benders cuts) applied to all stage problems. This computational trick ensures reasonable reservoir trajectories in the first iteration. To guarantee convergence of the cost gap, these end-values should be removed from all but the last stage problems in the subsequent iterations.

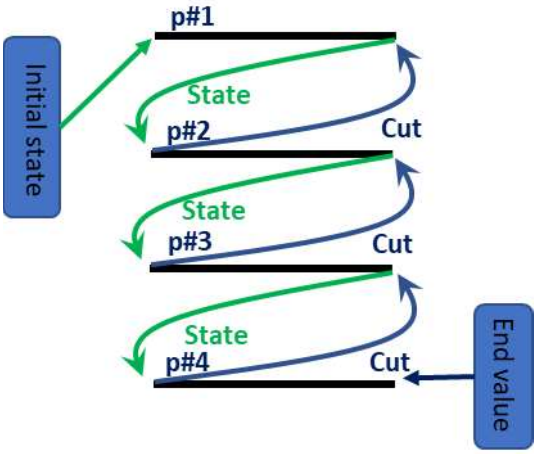

Figure 16 Serial execution of temporal decomposition.

3.3 Computational Experiments

The computational study was carried out for a weekly decision problem with hourly time resolution representing weeks 9 and 31 for weather years 1969 (little inflow) and 2013 (medium inflow). We divided the scenario LP problems into 2,4,7 and 168 subproblems. The results presentation emphasizes on convergence of the cost gap, simulated power prices and some selected reservoir trajectories. All results are compared to solving the full LP problem, i.e., without decomposition. The case matrix is shown in Table 2.

Table 2 Case matrix

Case no	Week	Weather year	NSub
1	9	1969	2
2	9	1969	4
3	9	1969	7
4	9	1969	168
5	31	2013	2
6	31	2013	4
7	31	2013	7
8	31	2013	168

A maximum number of 20 iterations was allowed in the tests. Note that we ran additional experiments with >> 20 iterations to check the convergence properties. It was found that convergence towards a

negligible cost gap was a slow process often requiring a few hundred iterations. For the test cases presented here, going beyond 20 iterations was not considered computational efficient when compared to solving the full LP problem.

Although the cases are similar to those presented earlier, we note that the minimum cost estimates differ since adjustments in transmission capacities and penalties were made.

The full LP problem has 1.35 million variables and 550 thousand constraints and requires 105 seconds for week 9 and 300 seconds for week 31 using the dual simplex algorithm (fastest). This problem covers the basic constraints:

- Power balances per area and time step
- Reservoir balances per module and time step
- Benders cuts constraining future expected cost

3.3.1 Convergence Properties

The decomposition algorithm converges when the upper and lower cost bounds are within a predefined tolerance. From Figure 17 and Figure 18 we observe that the cases with lower number of stages (labelled N in the figures) seem to converge reasonably well within the maximum number of iterations. It was verified that all cases converge to objective value of the full problem when a sufficient number of iterations is allowed.

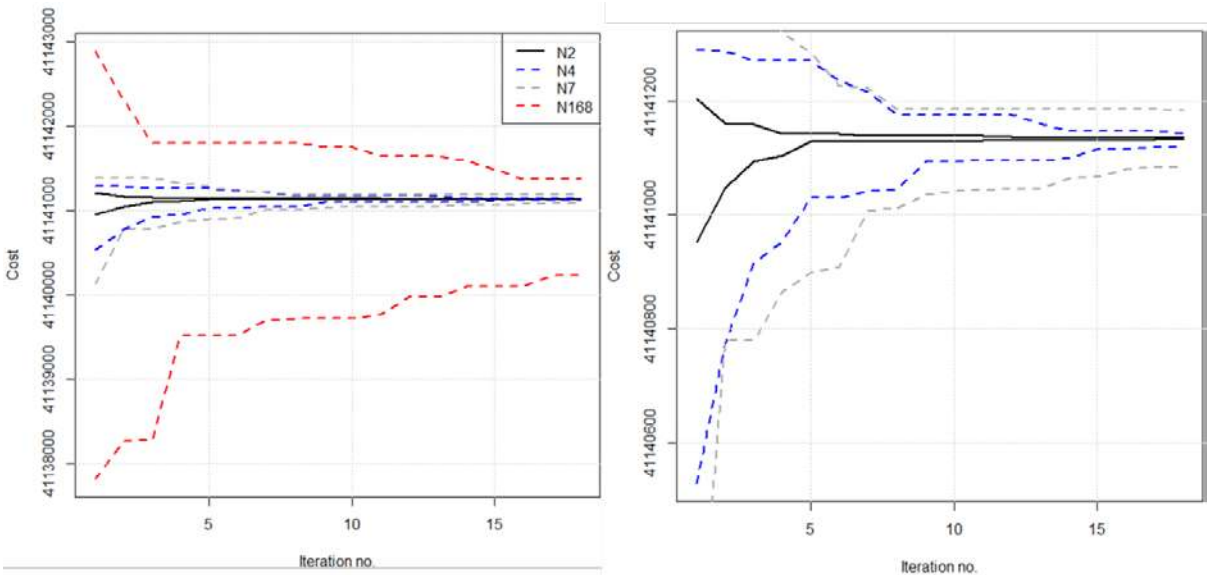

Figure 17 Upper and lower boundaries for cases 1-4 for week 9. Left figure zooms in.

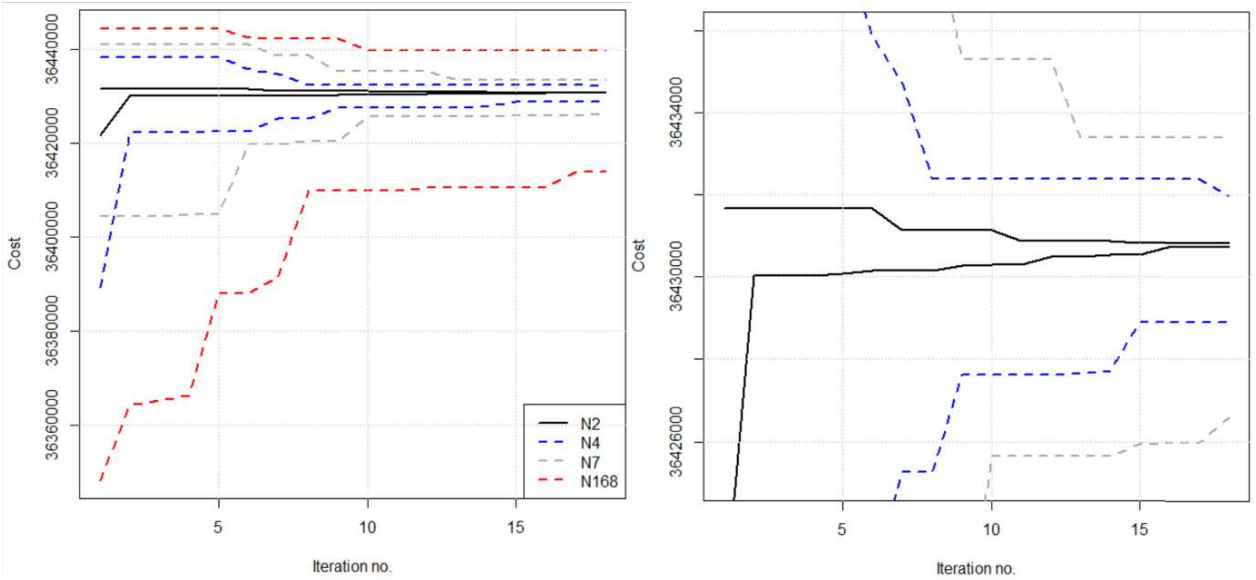

Figure 18 Upper and lower boundaries for cases 5-8 for week 31. Left figure zooms in.

Figure 19 shows the accumulated time spent by CPLEX for the different cases. There seems to be some effect of warm start when NSub=2, explaining the concave shape of the black, solid-drawn curve. This effect is not as pronounced for larger values of NSub. The accumulated time is notably faster for NSub =168, but one should keep in mind the high cost gap for this value of NSub after 20 iterations, indicating a poor solution quality.

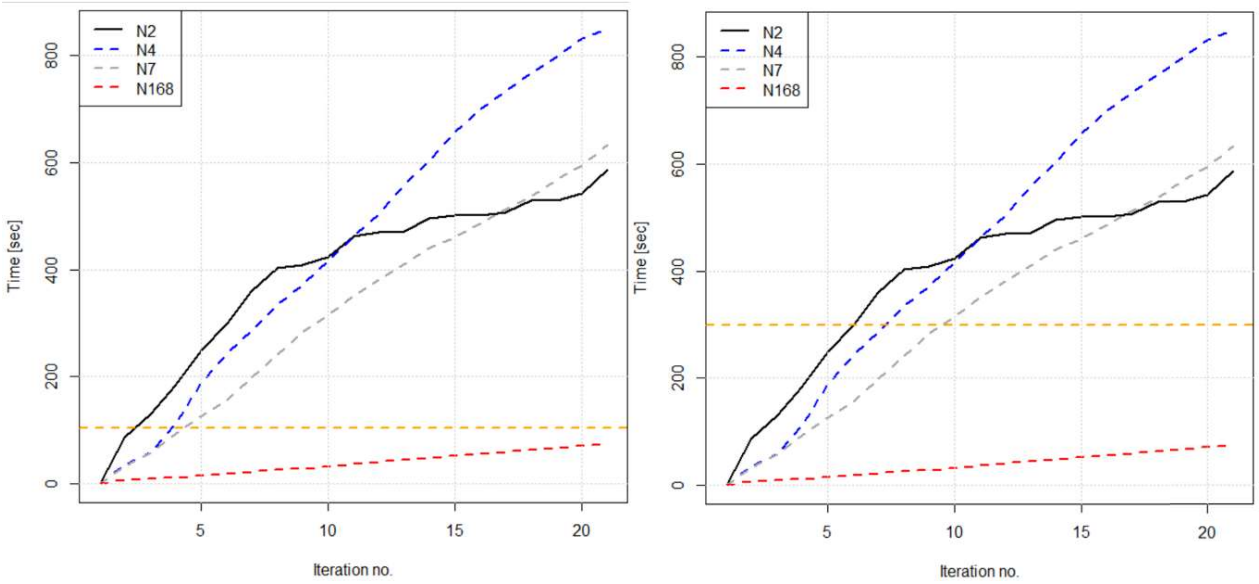

Figure 19 Accumulated time spent by the optimization solver for week 9 (cases 1-4, left) and week 31 (cases 5-8, right). The orange lines indicate the time for solving the full LP problem.

For comparison, the computational times for one PRIBAS iteration are shown in Table 3. Clearly, this heuristic is faster solving the decomposed problem, but comes at the cost of lower precision in simulation results when compared to the solution of the full LP problem.

Table 3 Computational time (in seconds) for one PRIBAS iteration.

	N2	N4	N7	N168
Week 9	44	39	35	9
Week 31	61	50	47	9

3.3.2 Simulated Power Prices

The simulated power prices for 2 different areas (OSTLAND and SVER-ON1) are shown in the following figures. Prices are extracted after the first iteration (a PRIBAS iteration) and after the last iteration (iteration 20) and compared with those obtained when solving the full problem.

With larger subproblem sizes (or low NSub) the price patterns are closer to the full LP solution after the maximum number of iterations. For cases 4 and 8 with NSub=168, prices from both the PRIBAS iteration and the decomposition deviate significantly from prices from the full LP solution. This is particularly pronounced in case 8, shown in Figure 23. As seen in the previous section, 20 iterations are too few to arrive at an acceptable convergence gap when NSub=168, and consequently the prices have not stabilized.

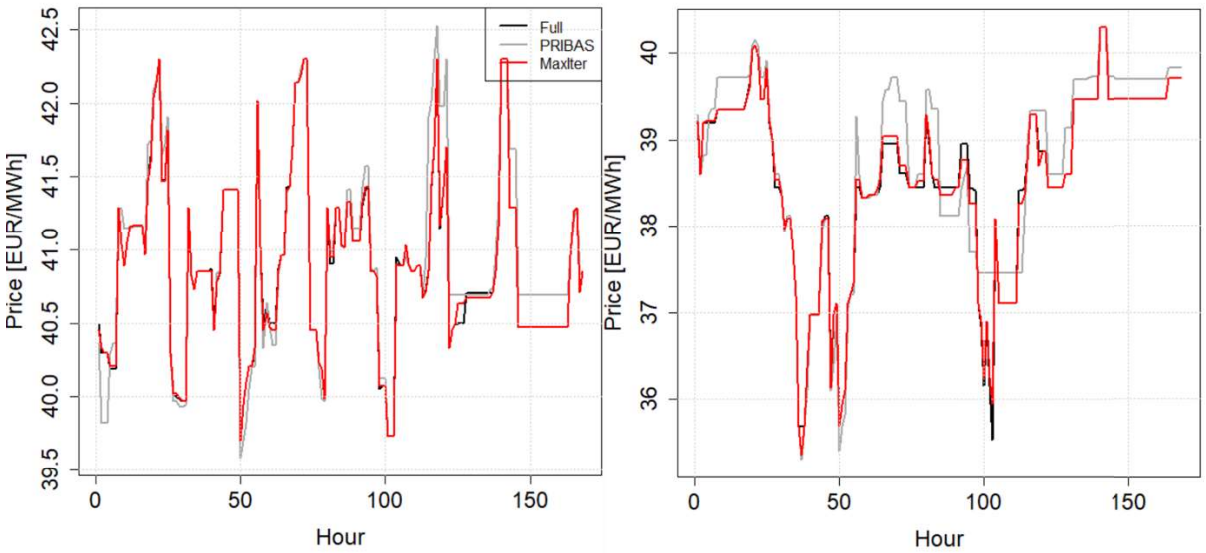

Figure 20 Prices in areas 1 (OSTLAND) and 12 (SVER-ON1) for case 1 (week9, NSub = 2).

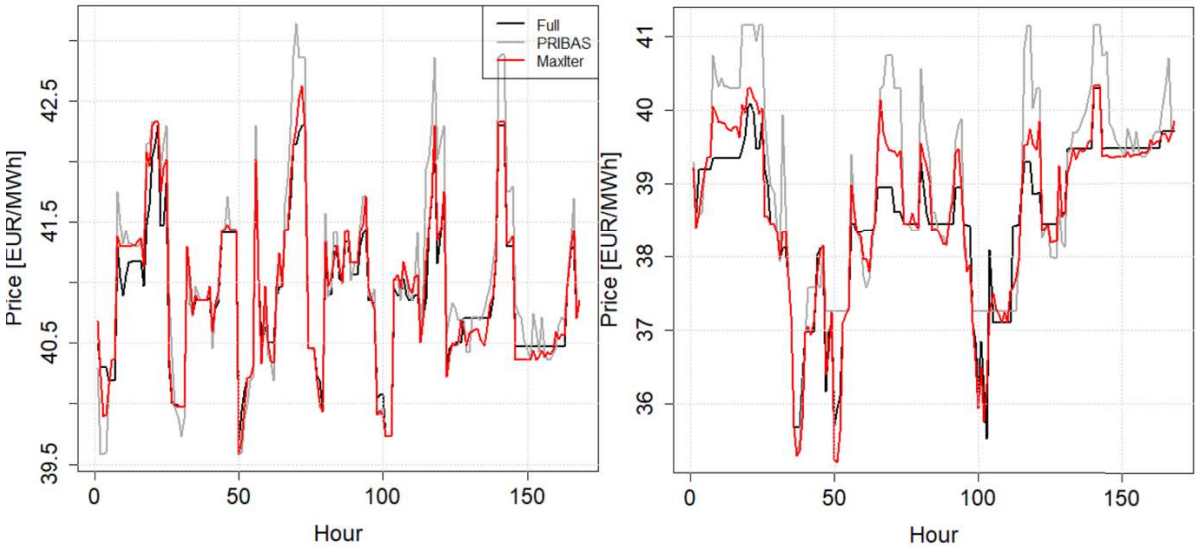

Figure 21 Prices in areas 1 (OSTLAND) and 12 (SVER-ON1) for case 4 (week9, NSub = 168).

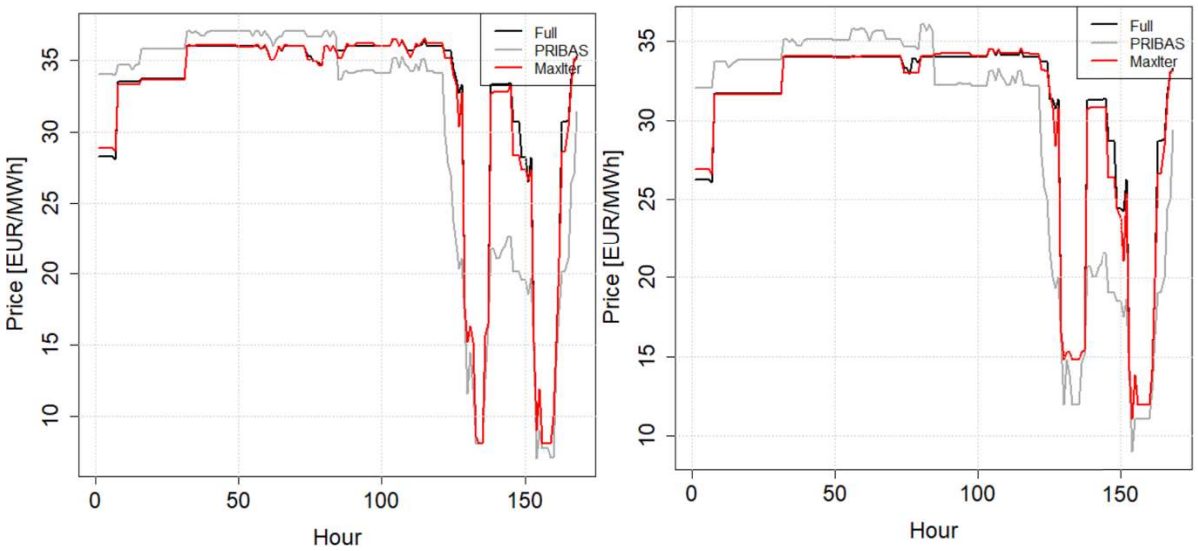

Figure 22 Prices in areas 1 (OSTLAND) and 12 (SVER-ON1) for case 5 (week31, NSub = 2).

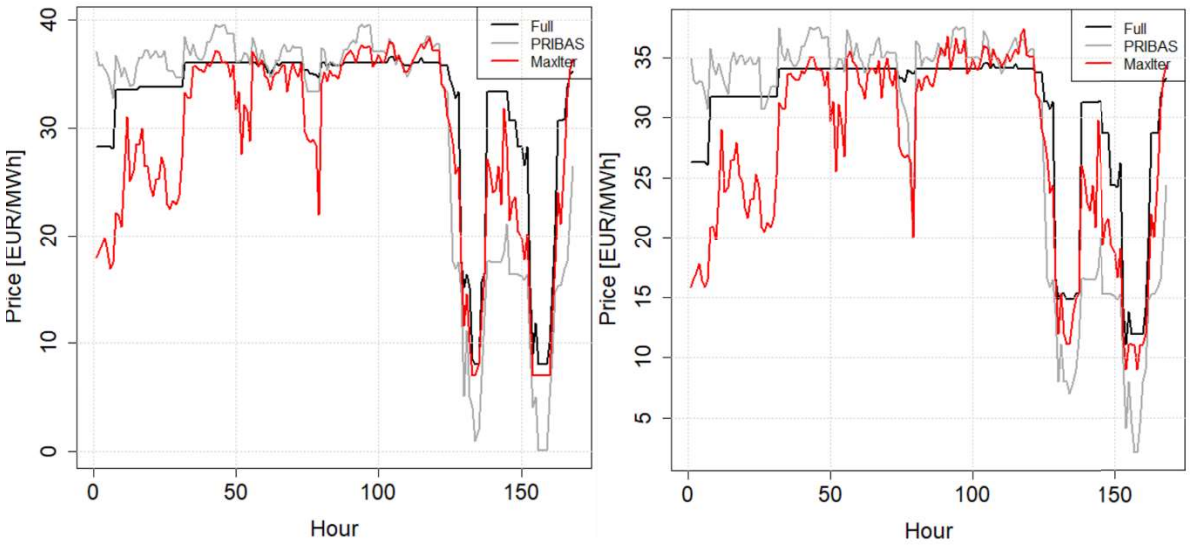

Figure 23 Prices in areas 1 (OSTLAND) and 12 (SVER-ON1) for case 8 (week31, NSub = 168).

3.3.3 Reservoir Volumes

We selected four reservoirs that differ in storage volume and regulation capability and present their calculated reservoir operation in the following. Figure 24 and Figure 25 presents volume trajectories obtained in case 1 for the two large reservoirs Blåsjø and Møsvatn and the two smaller reservoirs Vessingsjø and Vrenga, respectively. We note that the trajectories obtained from the full LP and from the last iteration coincide, while the PRIBAS iteration deviates slightly. The differences are more pronounced in case 4, as shown in Figure 26 and Figure 27.

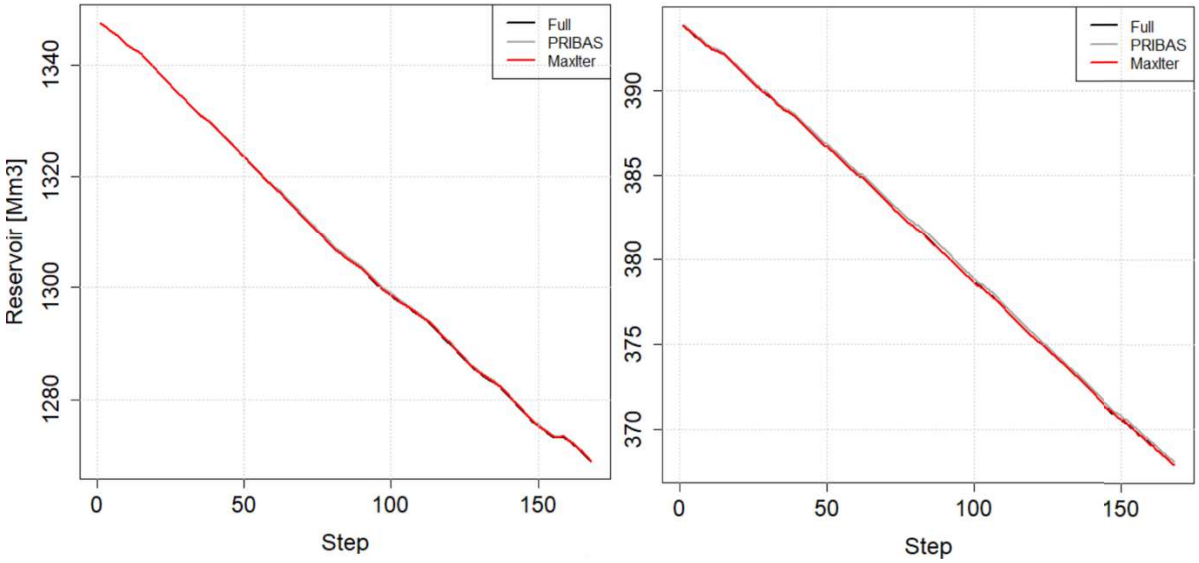

Figure 24 Reservoir volumes for Blåsjø and Møsvatn for case 1 (week9, NSub = 2).

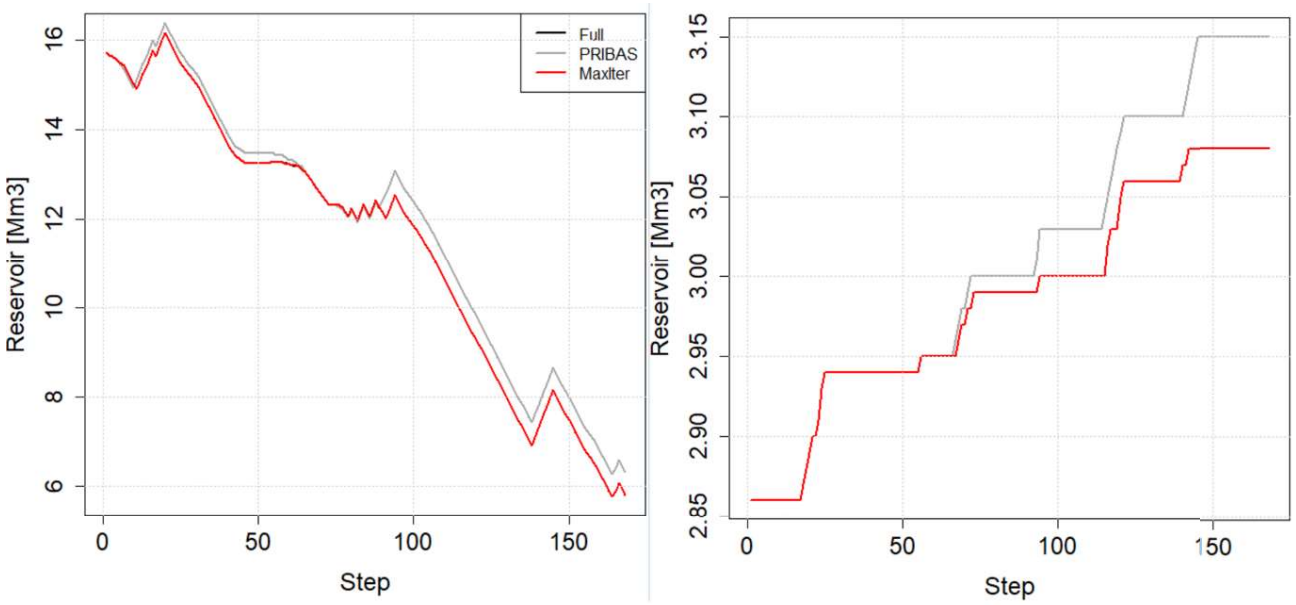

Figure 25 Reservoir volumes for Vessingsjø and Vrenga for case 1 (week9, NSub = 2).

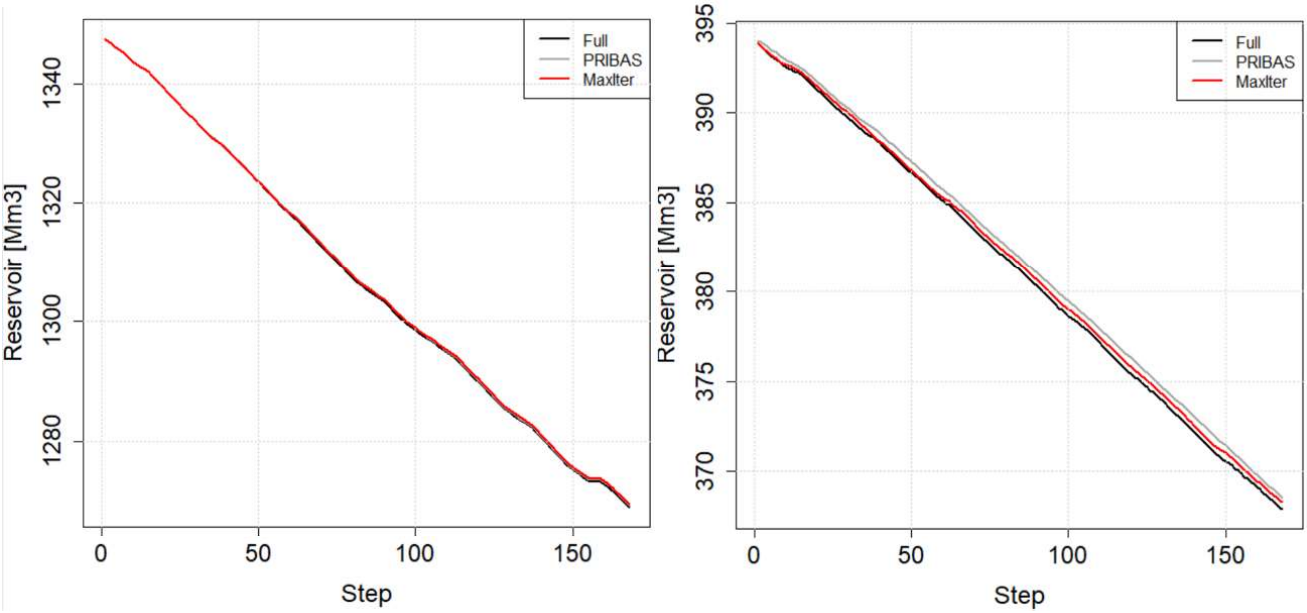

Figure 26 Reservoir volumes for Blåsjø and Møsvatn for case 4 (week9, NSub = 168).

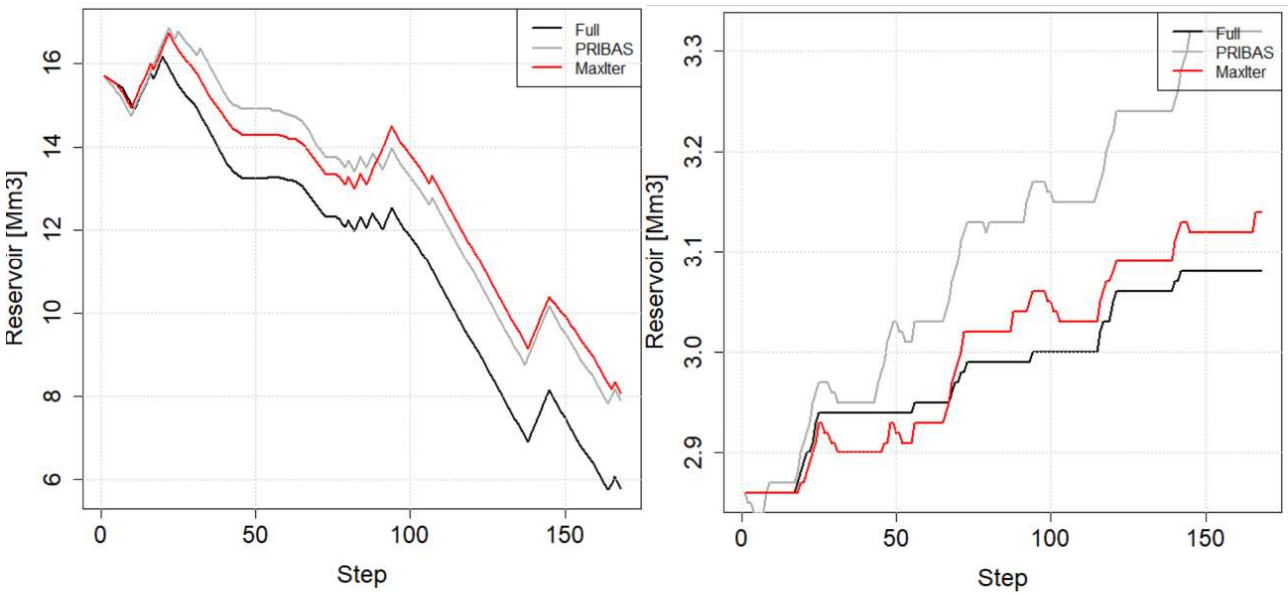

Figure 27 Reservoir volumes for Vessingsjø and Vrenga for case 4 (week9, NSub = 168).

3.4 Computational Experiments – Additional Time Linking Constraints

Results presented so far are from LP problems covering the basic functionality. They do for instance not include additional features such as power flow constraints, start-up costs on generation units, and limited ramping capability on generation, discharge or power flows.

We added constraints for limiting the ramping capability on all 100 transmission lines in all 168 time steps, i.e., an additional 168.000 constraints. Constraints are modelled as follows:

$$-\Delta_l \leq f_{lk} - f_{l,k-1} \leq \Delta_l \tag{1.10}$$

Where f_{lk} is the flow, Δ_l the maximum ramp up/down, index *l* indicates line number and *k* time step number. The additional time coupling in (1.10) was accounted for in the Benders cuts. Thus, the new cut has two types of state variables; reservoir volumes and line flows. The initial flow states were considered "free", e.g., as decision variables in the optimization.

We verified that the decomposition algorithm converges to the same optimal solution as the full LP problem. Tests were conducted using the same case matrix as in Table 2. Ramping constraints were considered on all transmission lines (both AC and HVDC) allowing ramping capabilities of 100%, 80%, 50% and 20% of the maximum line flow. 100% ramping capability corresponds to the solution of a problem without ramping constraints, e.g., ramping constraints are included in the LP model but not binding.

We seek answers to how the introduction of additional time-linking constraints impact the convergence properties and computation time.

3.4.1 Convergence Properties

Figure 28 and Figure 29 show how the cost gaps (upper bound minus lower bound) gradually decrease with increasing iteration number with different number of subproblems (NSub) and with different ramping capabilities for week 9 inflow year 1969.

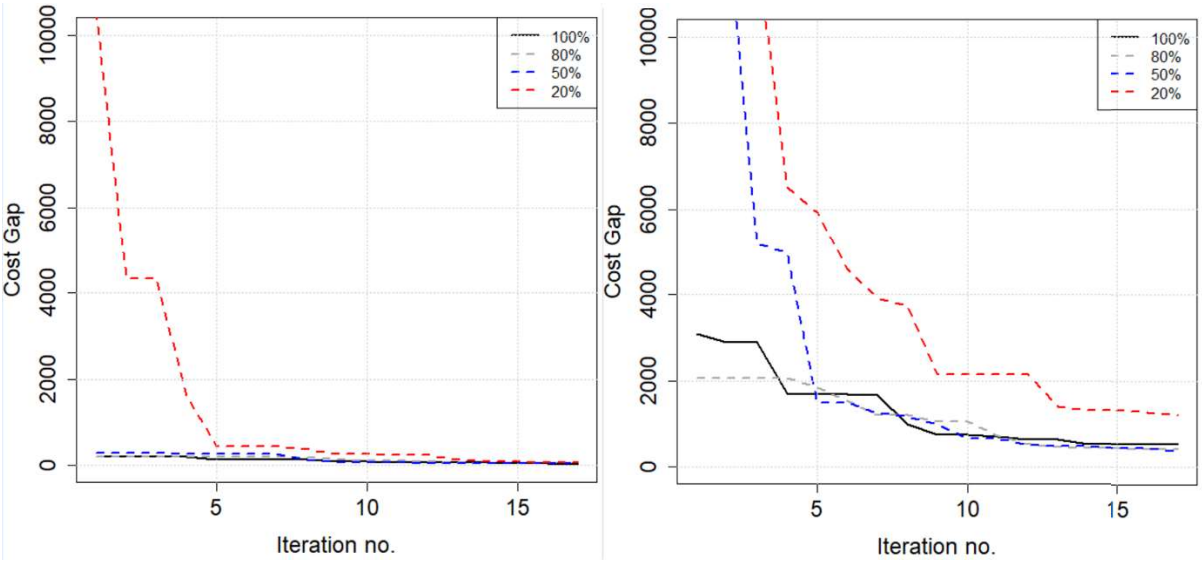

Figure 28 Cost gaps for various levels of ramping constraints for NSub=2 (left) and NSub=4 (right).

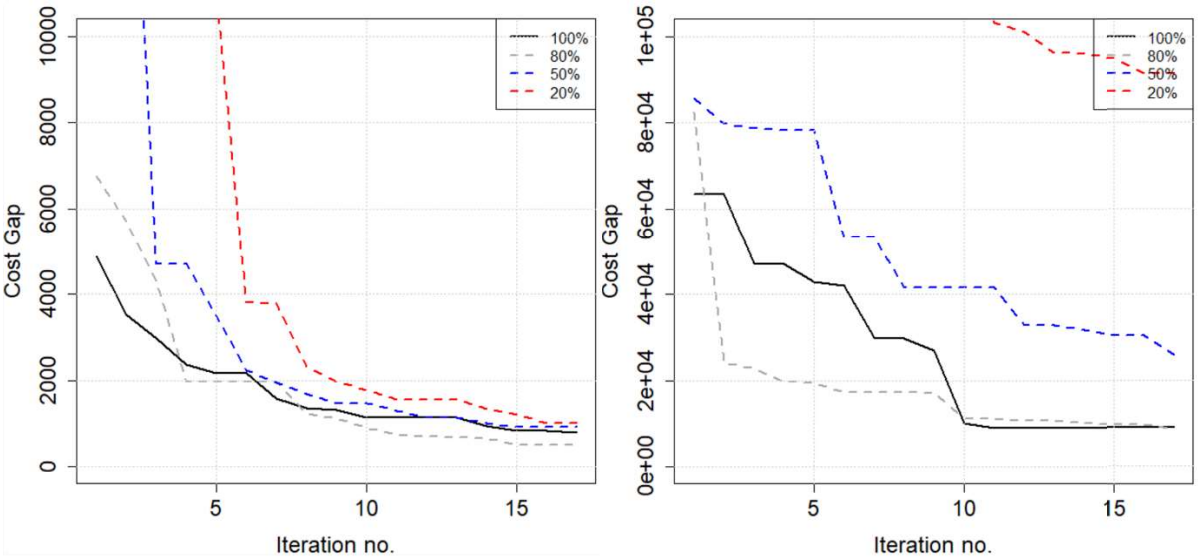

Figure 29 Cost gaps for various levels of ramping constraints for NSub=7 (left) and NSub=168 (right).

For all values of NSub we observe that a lower ramping capability leads to an increase in cost gap and thus a slower convergence. By lowering the ramping capability, we further constrain the optimization problem, and it becomes more difficult to find the optimal solution. Moreover, flow as a state variable is higher valued with decreasing ramping capability, which further challenges the decomposition process in correctly valuating the set of state variables throughout iterations.

3.4.2 Computation Time

Figure 30 and Figure 31 show the accumulated time per iteration when solving cases 1-4 with varying ramping capability. For the lowest ramping capability, 20% shown to the right in Figure 31, the solution time for the full LP problem is substantially higher (658 seconds) than the others (all less than 160 seconds). In this severely constrained case, we find that decomposition can to a larger extent assist in

speeding up computations. Obviously, the computation times should be evaluated together with the cost gaps and solution quality to decide on a suitable convergence criterion.

As for the previous tests without ramping constraints, the effect of warm start is more evident for NSub=2, indicated by the pronounced concavity of black curve in the figures. By further investigating the solution time of individual subproblems, it became evident that, although some of the subproblems benefit from warm start in the 20 iterations considered, others do not. The subproblem with the highest computation time tends to dominate the total computation time per iteration.

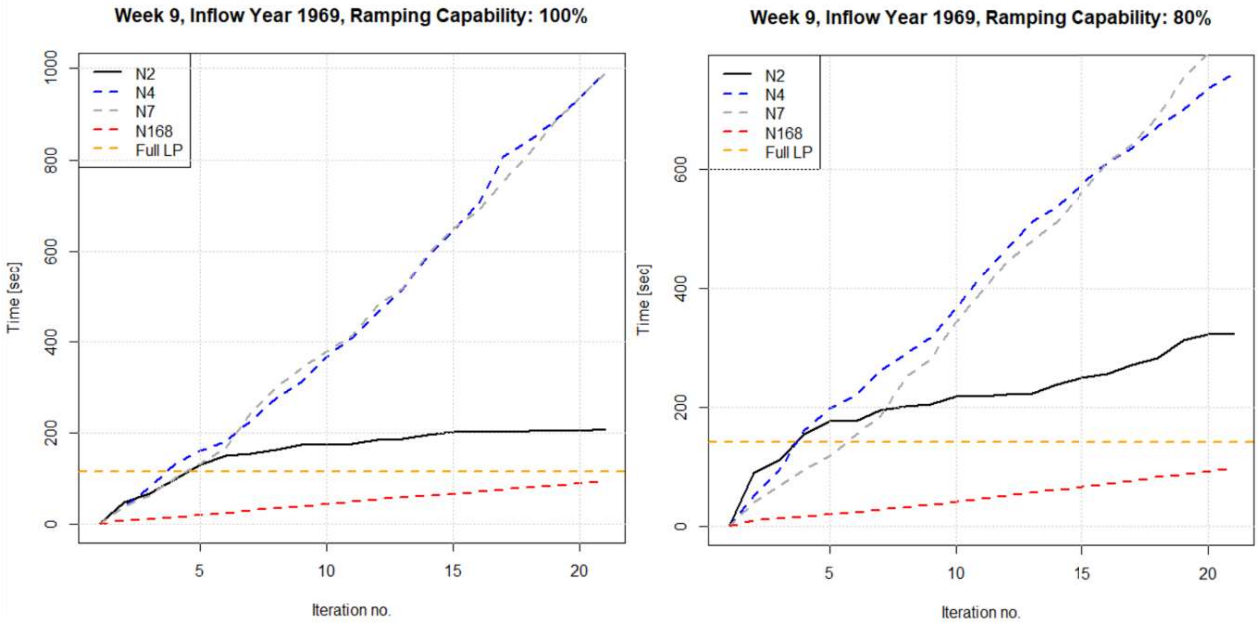

Figure 30 Accumulated time spent by the optimization solver for week 9 (cases 1-4) with 100% (left) and 80% (right) ramping capability. The orange lines indicate the time for solving the full LP problem.

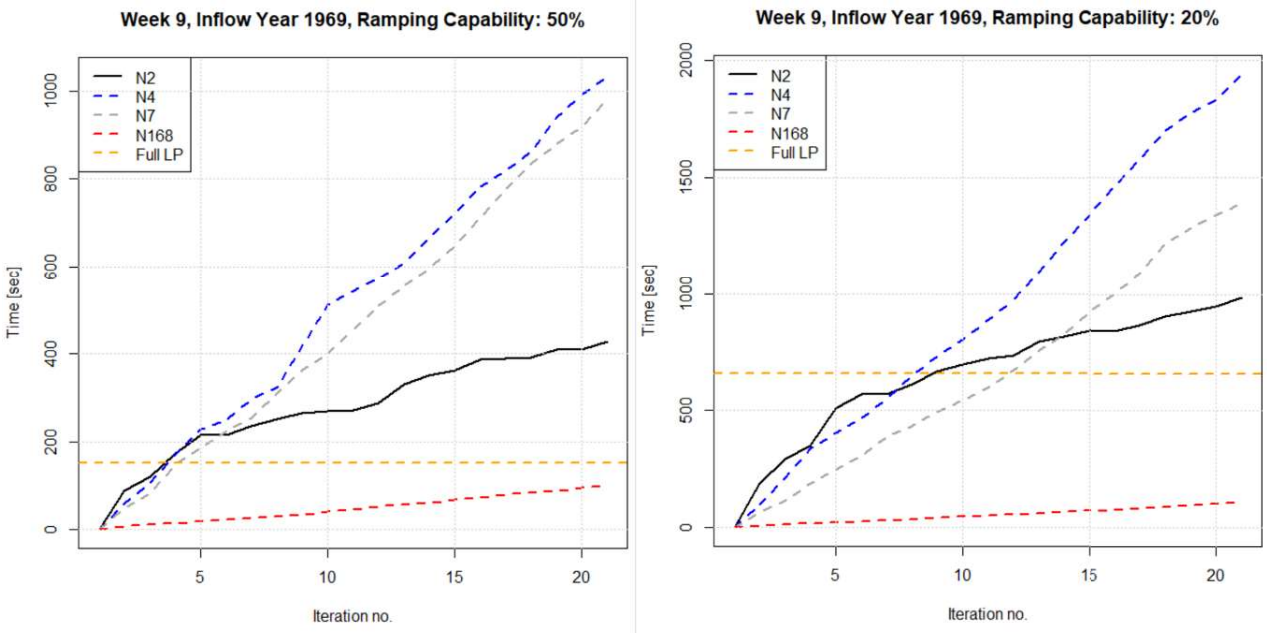

Figure 31 Accumulated time spent by the optimization solver for week 9 (cases 1-4) with 50% (left) and 20% (right) ramping capability. The orange lines indicate the time for solving the full LP problem.

For comparison, the computational times for one PRIBAS iteration for the cases with 20% ramping capability are shown in Table 3. The full problem is solved in 658 seconds for week 9 and 1031 sec for week 31. Clearly, this heuristic is faster solving the decomposed problem, but comes at the cost of lower precision in simulation results when compared to the solution of the full LP problem.

Table 4 Computational time (in seconds) for one PRIBAS iteration when ramping capability is 20%.

	N2	**N4**	**N7**	**N168**
Week 9	95	76	62	19
Week 31	196	196	97	10

The computational benefit of conducting a PRIBAS iteration rather than solving the weekly decision problem is substantial for the problem with tight ramping capability: It can be reduced to 10% when solving daily subproblems and 1% when solving hourly subproblems for week 31. However, the latter case do not find and optimal solution with respect to the ramping constraints.

4 Temporal Decomposition – Parallel Benders

Although the Benders decomposition scheme applied to a deterministic LP problem as described in the previous section is serial in nature, it is possible to exploit parallel processing to speed up computations. A scheme following the article [8] and described in Section 4.1 was implemented and tested as documented in Section 4.2.

4.1 A Parallel Processing Scheme

The concept is illustrated in Figure 32. Like the serial execution we allocate one processor (P#1-p#4) to each stage. The stage subproblem is built once at its respective processor benefits from a starting basis except from the first time it is solved.

In the parallel execution mode, the processors do not wait for all the former stages to compute a coherent trajectory (forward) and the cuts to recursively propagate (backwards). The notion of forward and backward iterations is therefore not relevant in the parallel mode, we can simply denote it as an iteration.

A *synchronous* parallel processing scheme was applied, where a single iteration provides a new state and a Benders cut from each stage problem. The synchronization forces all stage-problems to be solved before starting a new iteration. The synchronization is not strictly necessary but was found easier to implement. From a mathematical point of view the asynchronous execution does not violate the convergence guarantee provided by Benders decomposition, but one should expect a slower convergence rate since fully updated information is not waited for.

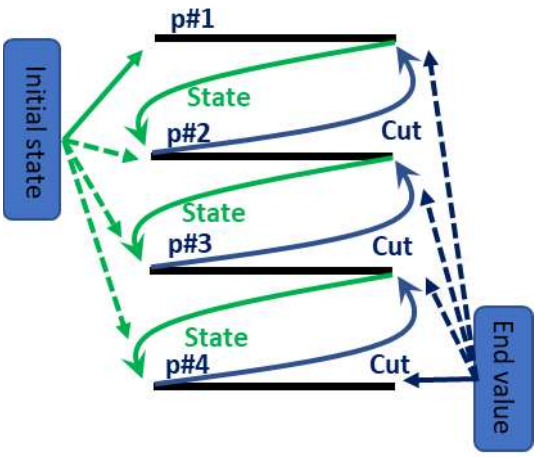

Figure 32 Parallel execution of temporal decomposition.

The Julia routine `pmap` was used to parallelize tasks. This routine will by default synchronize processes. More programming would be needed to facilitate an asynchronous execution, and this was not considered within the scope of this work.

The following subtasks are handled by `pmap`:

1) Build model (only first time)
2) Solve model (with starting basis from second iteration)

3) Update state trajectories and add cuts

A more detailed presentation of the synchronous parallel algorithm applied in this work is shown in Figure 33. The iterations are illustrated starting at the top line and proceeding downwards. We divide the scenario in 4 stages and assign a separate processor to each stage problem.

The initial state (reservoir volumes and others) should be given only in stage one, but we provide it to all stages in the first iteration as a heuristic to improve convergence. Similarly, end-value cuts should only be given to the last stage problem, but we provide it to all stage problems as a heuristic to avoid myopic solutions in the first iteration. These heuristics are not indicated in the figure.

Before starting a new iteration, each stage problem passes its final state to the next stage problem (illustrated by the green stapled lines) and a Benders cut (illustrated by the purple stapled lines) to the previous stage problem.

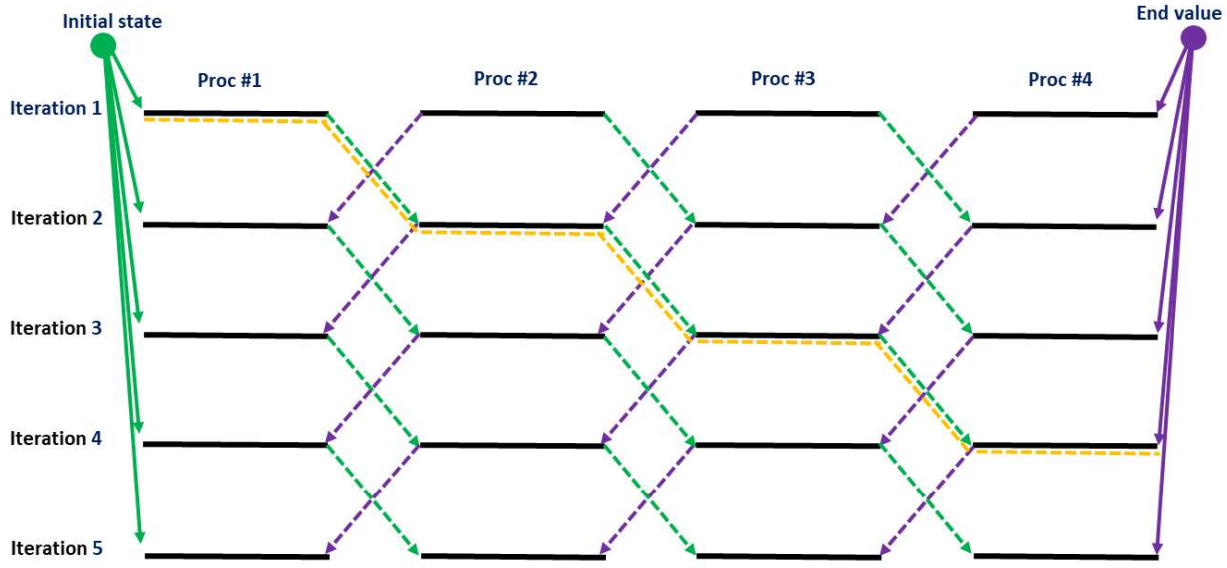

Figure 33 Illustrating the parallel processing scheme.

The lower cost bound is computed as the cost seen from stage 1 (as normal). The upper cost bound is normally computed by summing all costs per stage in the same iteration. However, this will lead to an inconsistent upper bound since the stage problems are not solved for the correct initial state (we did not wait for it). A more precise estimate is obtained when summing costs from the sequence of subproblems with corresponding initial states, as illustrated by the orange dotted line in Figure 33.

As mentioned above, we set up the parallel execution in a synchronous fashion, always waiting for each subproblem to solve completely. During experiments we observed that the LP subproblems differ significantly in solution time, especially in the later iterations when warm start is applied. While some subproblems takes less than 0.01 seconds to solve others may take tens of seconds. Obviously, waiting for the last subproblem to solve will hamper efficient use of computational resources in such cases, as the remining processes will be idle. Thus, we experimented with a heuristic to approximate an asynchronous implementation by allowing a maximum waiting time for each subproblem. If the problem was not solved within the time limit, the algorithm moves on using the previous cost estimates and state variables and

without creating a new Benders cut for that subproblem. This approach will often lead to slower convergence, but do not sacrifice the mathematical convergence guarantee.

4.2 Computational Experiments

A separate code for solving the parallel Benders decomposition scheme outlined above was developed. We refer to the synchronous parallel scheme as PAR-SYNC and the approximate asynchronous parallel scheme (wit a maximum time limit per subproblem) as PAR-ASYNC.

The four cases considered are presented in Table 5, all in the form of weekly decision problems. Although the cases are similar to those presented earlier, we note that the minimum cost estimates differ since adjustments in transmission capacities and penalties were made. It was chosen to test for 4 and 7 subproblems to adapt to the computational capability of the PC used (maximum 12 threads available).

Table 5 Case matrix.

Case no	Week	Weather year	NSub
1	9	1969	4
2	9	1969	7
3	31	2013	4
4	31	2013	7

Upper and lower bounds obtained per iteration are displayed in Figure 34 and Figure 35. We let the serial decomposition run for 20 iterations on the same problem and serve as benchmark. The two parallel decomposition schemes reached a similar cost gap for cases 1, 2 and 4 after approximately 60 iterations, while the gap in case 3 is wider with PAR-ASYNC after 60 iterations. In the case of the PAR-ASYNC scheme a time limit of 3.0 seconds was imposed per subproblem, made effective after the 7th iteration to allow the subproblems in the first 7 iterations to solve to completion.

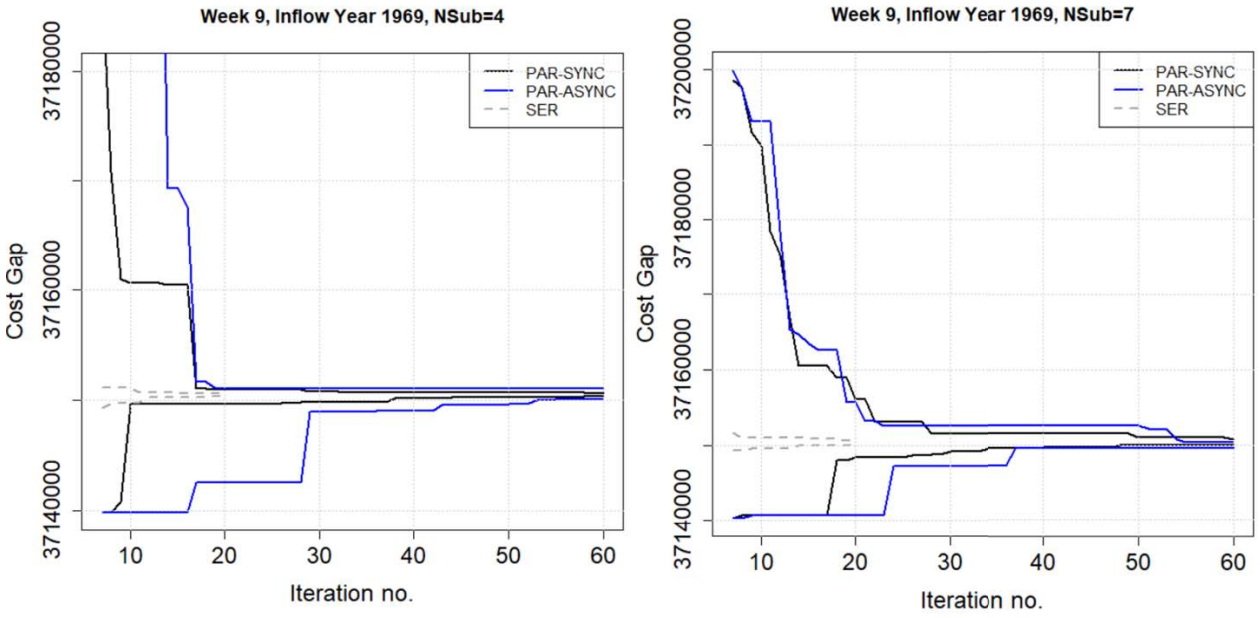

Figure 34 Convergence of cost gap for cases 1 and 2. Left: NSub=4, Right: NSub=8.

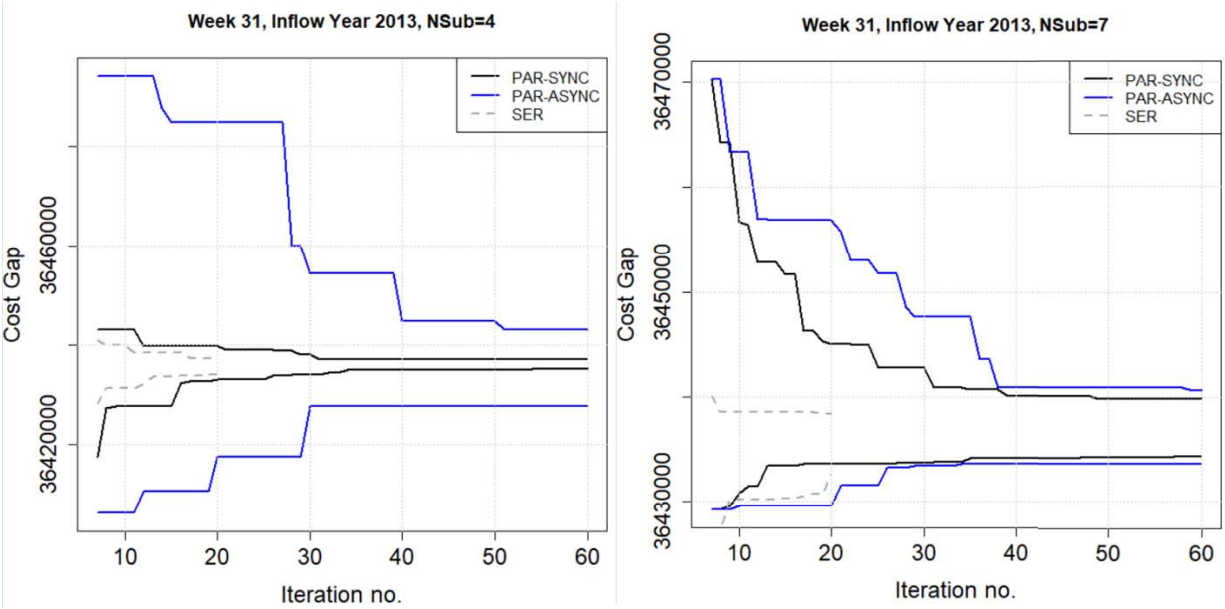

Figure 35 Convergence of cost gap for cases 3 and 4. Left: NSub=4, Right: NSub=8.

Accumulated time consumption spent by CPLEX is shown in Figure 36 and Figure 37. There is a substantial improvement in computation time with the synchronous parallel algorithm (PAR-SYNC) compared to the serial one, even when accounting for the fact that the parallel needs approximately 3 times more iterations to reach a similar cost gap. Still, the solution time of the full LP problem justify no more than 10-15 iterations with PAR-SYNC. The PAR-ASYNC scheme provide a significant speedup compared to PAR-SYNC, and for cases 3 and 4 is faster conducting 60 PAR-ASYNC iterations than solving the full LP problem. We also observe that the difference between accumulated time for PAR-SYNC and PAR-ASYNC is larger when NSub=7 than for NSub=4. This can be explained by the fact that NSub=4 leads to larger subproblems that are likely to often take more than the PAR-ASYNC maximum limit of 3 seconds to solve.

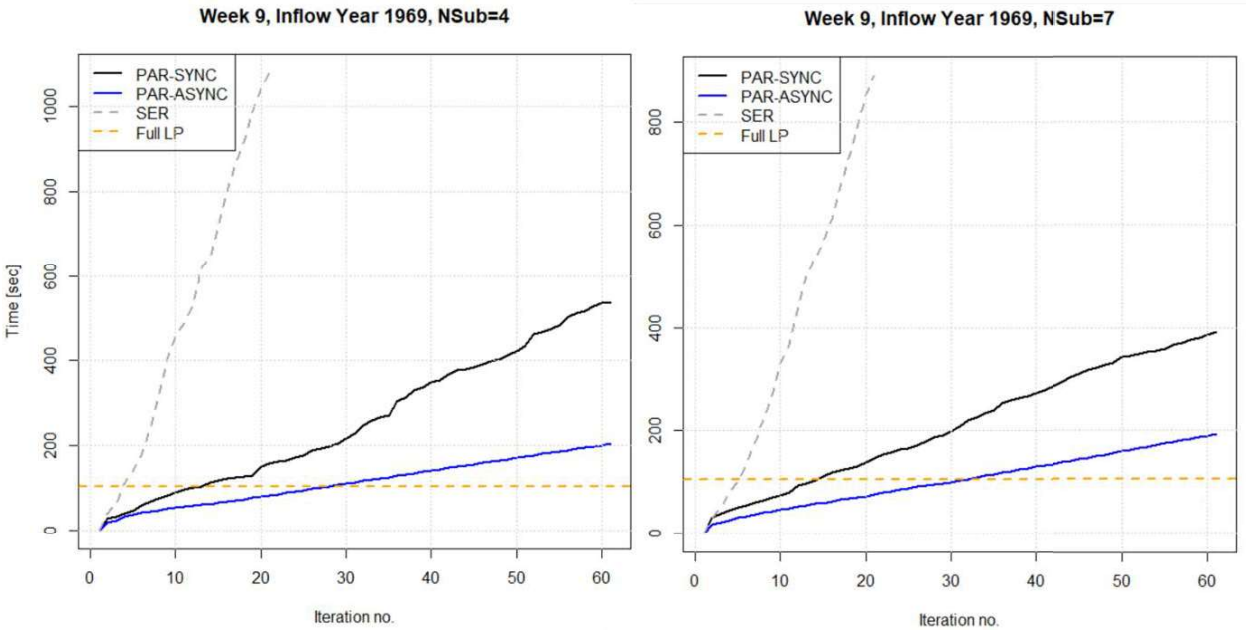

Figure 36 Accumulated time spent by the optimization solver for week 9 (cases 1-2) with NSub=4 (left) and NSub=7 (right). The orange lines indicate the time for solving the full LP problem (105 sec).

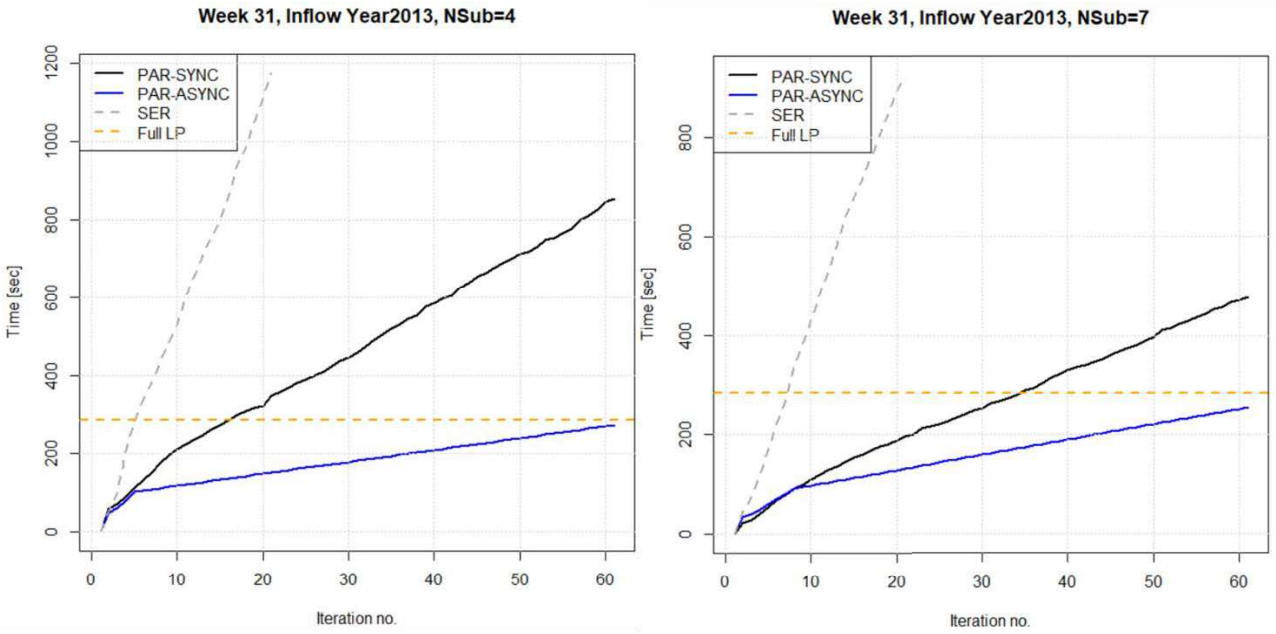

Figure 37 Accumulated time spent by the optimization solver for week 31 (cases 3 and 4) with NSub=4 (left) and NSub=7 (right). The orange lines indicate the time for solving the full LP problem (285 sec).

5 Discussion and Conclusions

Below is our summary of the main findings from the computational experiments described in this report and a high-level assessment of the feasibility of and potential gain from further decomposition of the Fansi model concept.

5.1 Findings from the Computational Experiments

We have tested a set of techniques for speeding up computation time of the hydrothermal scheduling problem formulated as a deterministic LP. These techniques were chosen based on early experiments and our best guesses on what is needed to further speed up the solution of both the master and scenario problems in Fansi.

Initially, we investigated the impact of relaxing reservoir constraints. Reservoir constraints are built per hydropower module and per time step per default in Fansi. Problems are typically defined for system representations with > 1000 modules and > 100 time steps, making this type of constraint dominant. We exploit the fact that many reservoirs do not operate close to their boundaries and are thus not binding in the LP problem solution. This is done by first relaxing intra-week reservoir constraints, and then iteratively solving the relaxed LP problem and adding those constraints that are violated. It was found that the major impact of reservoir constraints on the LP problem solution was covered in the first iteration. Moreover, by far most binding constraints are on small reservoirs. Thus, we suggested a heuristic involving a threshold for which reservoirs with higher maximal volumes initially have their reservoir constraints relaxed combined with a single iteration, adding violated constraints. This heuristic method provided high accuracy and significant computational savings compared to solving the full LP problem. Finally, we note that the impact of relaxing reservoir constraints depends on the state of the reservoirs and the length of the scenarios being considered. The conclusion from these investigations is clear:

Relaxation of reservoir constraints provides significant computational speedup with modest impact on solution quality.

Secondly, we investigated the potential for saving computation time by applying temporal decomposition of the hydrothermal scheduling problem formulated as a deterministic LP. Initial testing indicated that the potential is higher on scenarios of shorter time horizon, because the storage levels of larger reservoirs are subject to relatively small changes. Thus, all tests were conducted on LP problems with a weekly time horizon. The temporal decomposition is sequential in nature, meaning that one has to solve each subproblem in sequence. The first tests with serial Benders decomposition show that many iterations are required to meet a standard convergence criterion, and that the overall computation time exceeds the solution time for the full LP problem without decomposition. On the other hand, if one allows the cost gap remaining after a limited number of iterations, computational savings can be achieved with modest reduction in solution quality. In particular, the so-called 'PRIBAS iteration' provide promising results and is worth further exploration. We emphasize that the PRIBAS iteration only makes sense when applied to the master problem in Fansi (obtaining primal results) and not for the scenarios (obtaining dual results).

The PRIBAS iteration provides an interesting heuristic for decomposing the Fansi master problem. It allows for significant computational speedup, and leads to differences in results, particularly for the operation of small-scale storages.

Finally, the temporal decomposition algorithm was arranged in parallel according to a synchronous parallel processing scheme known from the literature. Parallel processing allowed breaking the sequential nature of the temporal decomposition algorithm and led to notable improvements in computation time. Through parallelization, the computation time per iteration was severely reduced on the one hand, while a higher number of iterations were needed to achieve a target cost gap on the other. Overall, the computational time was improved compared to the serial decomposition algorithm, but obviously the improvement comes at the cost of additional use of computational resources. To further improve the parallel implementation, we experimented with an approximate asynchronous parallel processing scheme were subproblems were allowed a maximum time for reaching a solution, and in the case of no solution, their "contributions" were not updated. This scheme showed promising results and could be worthwhile further investigations.

Asynchronous parallel processing of the Benders decomposition algorithm provided promising results for difficult LP problems and is worthwhile further investigations.

5.2 Further Temporal Decomposition in Fansi

Below is a list of pros and cons for further decomposing the SFP solved in Fansi along the time axis:

Pros:

- Temporal decomposition is already a part of the Fansi concept, and is relatively straightforward to implement and maintain. In contrast, spatial decomposition (e.g. by Lagrangian Relaxation) involves a deeper integration in the problem structure and may need additional functionality to recover primal results.
- There is no need for tailor-made model builders. All optimization problems (subproblems) cover the same system and system boundary.
- Techniques for decomposition in time using parallel processing has been studied in the existing literature and applied to similar type of problems with success [8].

Cons:

- The algorithm described for solving deterministic LP problems by decomposing along the time axis is sequential in nature. This is different from the spatial decomposition which is more intuitively suited for parallel computations.
- The importance of representing temporal couplings is likely to increase in the future power market models. Functionalities such as ramping (power flows, generation, water discharges, etc.), start-stop costs on power generation units, time-delays in rivers and end-user flexibility across time-scales all represent temporal couplings. Ideally, all such couplings should be represented in the Benders cuts, but from a practical point of view it is worth asking if the improved accuracy is worth the additional computational complexity.

6 References

[1] A. Helseth, B. Mo, A. L. Henden and G. Warland, "Detailed Long-Term Hydro-Thermal Scheduling for Expansion Planning in the Nordic Power System," *IET Generation, Transmission and Distribution,* vol. 12, pp. 441-447, 2018.

[2] J. R. Birge and F. Louveaux, Introduction to Stochastic Programming, 2 ed., Springer, 2011.

[3] O. M. Hansen, "Solution of the Economic Dispatch Problem by Spatial decomposition," SINTEF Energi , Trondheim, 2022.

[4] A. Helseth, S. Jaehnert and A. L. Diniz, "Convex Relaxations of the Short-Term Hydrothermal Scheduling Problem," *IEEE Transactions on Power Systems,* vol. 36, pp. 3293-3304, 2021.

[5] A. Helseth, "Spatial Benders decomposition for hydrothermal scheduling – Application to the Fansi market model," SINTEF Energi, Trondheim, 2022.

[6] A. Helseth and B. Mo, "Hydropower Aggregation by Spatial Decomposition – An SDDP Approach," *IEEE Transactions on Sustainable Energy,* 2022, DOI:10.1109/TSTE.2022.3214497.

[7] L. E. Schäeffer, B. Mo and I. Graabak, "Electricity Prices and Value of Flexible Generation in Northern Europe in 2030," in *13th International Conference on the European Energy Market*, Ljubljana, 2019.

[8] T. N. Santos, A. L. Diniz and T. C. L., "A New Nested Benders Decomposition Strategyfor Parallel Processing Applied to theHydrothermal Scheduling Problem," *IEEE Transactions in Smart Grid,* vol. 8, 2017.

[9] A. Helseth, M. Haugen, H. Farahmand, B. Mo, S. Jaehnert and I. Stenkløv, "Assessing the benefits of exchanging spinning reserve capacity within the hydro-dominated Nordic market," *Electric Power Systems Research,* vol. 1999, 2021.

[10] A. Helseth, "Stochastic network constrained hydro-thermal scheduling using a linearized progressive hedging algorithm," *Energy Systems,* vol. 7, pp. 585-600, 2016.