Parallel Dual Dynamic Integer Programming for Large-Scale Hydrothermal Unit-Commitment

Bruno Colonetti, Erlon Finardi, Samuel Brito, and Victor Zavala

Abstract—**Unit commitment has been at the center of power system operation for well over 50 years. Yet, this problem cannot be considered solved due to its size and complexity. Today, operators rely on off-the-shelf optimization solvers to tackle this challenging problem, and often resort to simplifications to make the problem more tractable and solvable in a reasonable time. Nonetheless, despite the simplifications and advancements in commercial optimization solvers, solving the unit commitment in a timely manner is still difficult. Thus, in this work, we propose a parallel dual dynamic integer programming approach for solving this challenging problem. Different from what can be currently found in the literature, our parallel approach is applied to a deterministic problem and thus requires induced parallelization. Our strategy is assessed on 20 cases of a system with over 7,000 buses and it is able to solve all instances to a 0.1% gap in less than two hours with speed-ups up to 9.2 compared to a sequential strategy, while the current time limit to the problem at hand is three hours for the same gap. The results show that the strategy enables the operator to solve more realistic problems within reasonable times.**

Index Terms—**dual dynamic programming, hydrothermal unit commitment, integer programming, parallelization.**

Nomenclature

Indices and Sets

$\mathcal{L}_b^+, \mathcal{L}_b^-$	Lines with positive flow in $(+)$ and out $(-)$ of bus b
$\underline{\mathcal{G}}_i, \overline{\mathcal{G}}_i$	Thermal units ($\mathcal{G}$) participating in the combined generation limit i
$\underline{\mathcal{I}}_t^{\mathrm{g}}, \overline{\mathcal{I}}_t^{\mathrm{g}}$	Upper (overline) and lower (underline) bounds on the combined generation of groups of thermal units
$b \in \mathcal{B}, g \in \mathcal{G}, h \in \mathcal{H}, l \in \mathcal{L}$	Buses, thermals units, hydro plants, transmission lines
$j \in \mathcal{A}$	Linear inequalities in the cost-to-go function
$j \in \mathcal{F}_h$	Linear inequalities of hydro plant h's approximation of its hydropower function
$t \in \mathcal{T}, \mathcal{T} = \{1, ..., \mathrm{T}\}$	Time periods in the planning horizon

Constants

$\mathrm{A}_j^{\mathrm{coeff}}, \mathrm{A}_j^{\mathrm{coeff}}$	Coefficient and constant of the cost-to-go function's j inequality (\$/hm^3, \$)
C^{h}	Flow-to-volume conversion ($1.8 \cdot 10^{-3}$ hm$^3\cdot$ s/m^3)
$\mathrm{F}^{\mathrm{vol}}, \mathrm{F}^{\mathrm{turb}}, \mathrm{F}^{\mathrm{spil}}, \mathrm{F}^{\mathrm{const}}$	Coefficients of the hydropower function's approximation: volume, turbine discharge, spillage, and constant (p.u./hm^3, p.u./(m^3/s), p.u./(m^3/s), p.u.)
I	Incremental inflow (m^3/s)
$\mathrm{L}_{b,t}$	Bus load (p.u.)
$\mathrm{SD}_{g,i}^{\mathrm{step}}$	ith step in the start-up trajectory. From 0 to $\mathrm{SD}_g - 1$
$\mathrm{SU}_g, \mathrm{SD}_g$	Periods in the start-up and shut-down trajectories
$\mathrm{SU}_{g,i}^{\mathrm{step}}$	ith step in the start-up trajectory. From 0 to $\mathrm{SU}_g - 1$
$\mathrm{T}_g^{\mathrm{up}}, \mathrm{T}_g^{\mathrm{down}}$	Minimum up and down times
$\underline{\mathrm{F}}, \overline{\mathrm{F}}, \mathrm{X}$	Bounds on the flows in transmission lines, and line reactance (p.u., p.u., p.u./rad)
$\underline{\mathrm{P}}_g^{\mathrm{t}}, \overline{\mathrm{P}}_g^{\mathrm{t}}$	Bounds on thermal generation (p.u.)
$\underline{\mathrm{P}}_h^{\mathrm{h}}, \overline{\mathrm{P}}_h^{\mathrm{h}}$	Bounds on hydro $^{\mathrm{h}}$ generation $^{\mathrm{t}}$ (p.u.)
$\underline{\mathrm{R}}_g, \overline{\mathrm{R}}_g$	Ramping limits (p.u./(30 min))

Hydro variables

α	Estimate future cost (\$)
$p^{\mathrm{h}}, d^{\mathrm{h}}$	Generation (p.u.), Status of the power plant (0/1)
$p_{h,j,t}^{\mathrm{hpf}}$	Generation output associated with plane j of plant h in time t (p.u.)
v, q, s	Reservoir volume, turbine discharge, and spillage (hm^3, m^3/s, m^3/s)
$v^{\mathrm{out}}, v^{\mathrm{in}}$	Water outflow and water inflow (hm^3, hm^3)
$v^{\mathrm{s,\ out}}, v^{\mathrm{s,\ in}}$	Slacks for the water balance (hm^3, hm^3)
$z_{h,j,t}^{\mathrm{hpf}}$	Binary variable associated with plane j of plant h in time t (0/1)

Network variables

θ, f	Bus angles and line flows (rad, p.u.)
$p^{\mathrm{s,\ n,\ +}}, p^{\mathrm{s,\ n,\ -}}$	Slacks variables: virtual generation, virtual load (p.u., p.u.)

Thermal variables

d^{t}	Status of the dispatch phase (0/1)
$p^{\mathrm{t}}, p^{\mathrm{t,\ disp}}$	Total generation (p.u.), and generation in dispatch phase (p.u.)
y, x	Start-up and shut-down decision (0/1)

Manuscript received April 19, 2005; revised August 26, 2015. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Award Number 001, and by Norte Energia through the Agência Nacional de Energia Elétrica (ANEEL) R&D Project PD-07427-0318/2018. (corresponding author: Bruno Colonetti.)

B. Colonetti and Erlon Finardi are with LabPlan, Department of Electrical and Electronic Engineering, Federal University of Santa Catarina, Florianópolis, Santa Catarina, 88040-900, Brazil. Erlon is also with INESC P&D Brasil, Santos, São Paulo 11055-300, Brazil. (e-mail: colonetti.bruno@posgrad.ufsc.br; erlon.finardi@ufsc.br)

Samuel Brito is with the Department of Computing and Systems, Federal University of Ouro Preto, Brazil. (e-mail: samuelbrito@ufop.edu.br)

Victor Zavala is with the Department of Chemical and Biological Engineering, University of Wisconsin – Madison, Madison, WI 53706, US. (e-mail: zavalatejeda@wisc.edu)

I. Introduction

Over the years, the management of electrical energy resources has seen significant changes and, because of recent technological developments and economic and environmental concerns, it continues to require intensive research. In particular, a problem that remains computationally challenging after more than 50 years of research is the unit-commitment (UC) problem. In its most complete form, this problem is a mixed-integer nonlinear optimization problem that is computationally intractable. As such, the UC problem is usually simplified to a mixed-integer linear programming (MILP) problem, both to facilitate its solution and also to comply with current market practices. However, the resulting MILP is still computationally challenging due to its size and combinatorial nature. Consequently, researchers have turned to decomposition approaches to overcome tractability limits. Among these techniques, dual dynamic programming (DDP) approaches provide a promising avenue. Perhaps the most used DDP algorithm is the so-called stochastic DDP (SDDP) method [1]. In summary, this consists of a stage-wise decomposition where the lower bound is obtained through the iterative improvement of recourse functions (the backward step), and the upper bound is statistically approximated (the forward step). Despite its success, one of SDDP's most apparent shortcomings is dealing with nonconvexities in the subproblems. Recently, SDDP has been extended to handle integer decisions and this approach is known as Stochastic Dual Dynamic Integer Programming (SDDiP) [2]. This approach has been applied to stochastic scheduling problems, [3] and [4], has sparked renewed interest in the application of DDP to nonconvex problems in the energy industry. As their names suggest, SDDP and SDDiP are used in stochastic problems, which have the characteristic of being relatively easily parallelizable. In a deterministic setting, such as the case of current real-life UC problems, SDDiP then assumes the name dual dynamic integer programming (DDiP). In this context, parallelism is not that easy to exploit: the method is essentially sequential and, in its classical form, parallelism in DDiP is restricted to low-level optimization. For situations where distributed computational resources are abundant, efficiently exploiting them is crucial for the success of a solution process.

The study of parallel DDP algorithms has been mainly restricted to convex multistage stochastic linear problems. In this context, multiple computational resources are exploited through either node or scenario parallelization, in either a synchronous or an asynchronous setting [5]. For instance, the parallel SDDP algorithm proposed in [6] distributes the computational load of the backward step among several processors by assigning to each process different subsets of the set of scenarios for each stage of the problem — scenario parallelization. A similar approach is used in the forward step, where processes are assigned to different subsets of hydrological scenarios. In their paper, the authors tackle a long-term hydrothermal scheduling problem and the temporal decomposition is stage-wise, i.e., each stage of the stochastic process represents a subproblem. A multistage stochastic integer programming problem is addressed in [7] through a parallel SDDiP. In [7], the computational loads of the forward and backward steps are distributed beforehand among a fixed set of processes who are then synchronously coordinated to solve the problem. A long-term hydrothermal scheduling problem is the problem of interest in [8] where an asynchronous parallel version of the SDDP algorithm where processes are assigned to single stages is devised. In their work, the authors propose delaying the improvement of the resource functions to avoid synchronization points. As the algorithm progresses, the recourse functions are enhanced leading to better solutions and improving the bounds until convergence is achieved.

Using the aforementioned parallelization strategies in a deterministic setting is not trivial. To circumvent this difficulty, authors have usually resorted to relaxing coupling constraints with Lagrangian relaxation and then solving the resulting subproblems in parallel. In [9], the authors decompose a deterministic multi-area power flow problem by applying variable splitting to the area-coupling variables. The area subproblems are then reformulated as dynamic programming subproblems, and parallelization is achieved by allowing areas to use coupling variables values of adjacent areas from previous iterations. Thus, once one solution for each area is obtained, the authors drastically reduce waiting times that are inherent to the serial algorithm. In contrast, [10] and [11] use a temporal decomposition to solve, respectively, a week-long stochastic economic-dispatch problem and a security-constrained unit commitment problem. Augmented Lagrangian relaxation is then applied to the coupling constraints of the subhorizon subproblems, and the authors use the auxiliary problem principle to solve its subproblems in parallel in a synchronous manner. In [12], the solution strategy of [10] and [11] is extended to include a learning-based algorithm used to select a single temporal decomposition based on the load profile of the underlying problem. On the other hand, [13] addresses a deterministic short-term scheduling problem for a hydrothermal system. The predecessor of [8] is [13], in which a parallel DDP method (in the paper called ”nested Benders decomposition”) is presented. In their strategy, an unique process is responsible for solving a single stage subproblem and communication of forward (primal) and backward (dual) solutions is performed asynchronously. As in [6], in [13], the same temporal decomposition is used for all processes.

Considering the works presently found in the literature, our contributions are twofold:

- A new parallelization scheme for the DDiP based on the coordination of different temporal decompositions. As the literature review suggests, and, to the best of our knowledge, no attempt has been made to parallelize a DDP algorithm, either for a deterministic or an uncertain problem, based on different temporal decompositions.
- Thorough assessment of the proposed method on several cases of a large system. Additionally, we provide insights into the inner working of the method through the analysis of several metrics of running times and duality gaps.

This paper is organized as follows. Section II gives the problem statement. Afterwards, in Section III we present our proposal. Then, Section IV introduces the cases used for assessing our strategy, while the results are given in Section V. Finally, our final remarks are presented in Section VI.

II. Problem statement

We are interested in solving a deterministic hydrothermal UC (HTUC) in a cost-based market. In this context, the system operator is responsible for determining the commitment (status ON/OFF) of generating units and their respective generation levels in order to meet the forecast load at minimum cost while satisfying local and system-wide constraints. In the following, we describe the model for each component of the HTUC.

A. Thermal generation model

In this model, a thermal unit can be in one of four states: off, in a start-up trajectory, in dispatch phase, or in a shut-down trajectory. The relations among d, y and x are given by the following constraints.

$$y_{g,t-\mathrm{SU}_g} - x_{g,t} - d^{\mathrm{t}}_{g,t} + d^{\mathrm{t}}_{g,t-1} = 0 \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{1}$$

$$\sum_{i=t-\mathrm{T}^{\mathrm{up}}_g}^{t} y_{g,i} \leq d^{\mathrm{t}}_{g,t} \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{2}$$

$$\sum_{i=t-\mathrm{T}^{\mathrm{down}}_g}^{t} x_{g,i} \leq (1 - d^{\mathrm{t}}_{g,t}) \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{3}$$

Eq. (1) are called logical constraints: they enforce that changes in d in consecutive periods result from start-ups or shut-downs. Eq. (2) and (3) are the classical minimum up-time and minimum down-time constraints. The bounds on the generation in the dispatch phase are imposed by Eq. (4), while Eq. (5) guarantee that the ramping limits are satisfied.

$$\underline{\mathrm{P}}^{\mathrm{t}}_g \cdot d^{\mathrm{t}}_{g,t} \leq p^{\mathrm{t,disp}}_{g,t} \leq \overline{\mathrm{P}}^{\mathrm{t}}_g \cdot d^{\mathrm{t}}_{g,t} \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{4}$$

$$\begin{aligned} -\underline{\mathrm{R}}_g \cdot d^{\mathrm{t}}_{g,t} - \underline{\mathrm{P}}_g \cdot (1 - d^{\mathrm{t}}_{g,t}) &\leq p^{\mathrm{t,disp}}_{g,t} - p^{\mathrm{t,disp}}_{g,t-1} \\ \leq \overline{\mathrm{R}}_g \cdot d^{\mathrm{t}}_{g,t-1} + \underline{\mathrm{P}}_g \cdot (1 - d^{\mathrm{t}}_{g,t-1}) &\quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \end{aligned} \tag{5}$$

The generation steps that must be followed during either the start-up trajectory or the shut-down trajectory are defined in Eq. (6). This equation has four terms: the generation of the unit, generation in the dispatch phase, the generation during start-up, and the generation during shut-down.

$$\begin{aligned} p^{\mathrm{t}}_{g,t} - p^{\mathrm{t,disp}}_{g,t} - \sum_{i=0}^{\mathrm{SU}_g - 1} (\mathrm{SU}^{\mathrm{step}}_{g,i} \cdot y_{g,t-i}) \\ - \sum_{i=0}^{\mathrm{SD}_g - 1} (\mathrm{SD}^{\mathrm{step}}_{g,i} \cdot x_{g,t-i}) = 0 \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \end{aligned} \tag{6}$$

Minimum generation for groups of thermal units are defined for each period t by the set $\mathcal{I}^{\mathrm{t,min}}_t$ in (7). Likewise, constraints (8) define upper limits to groups of thermal units over the period of the planning horizon. Note that, for both (7) and (8), we use slack variables, respectively, $p^{\mathrm{s,\,t,\,min}}_{i,t}$ and $p^{\mathrm{s,\,t,\,max}}_{i,t}$, to avoid the need of feasibility cuts.

$$\sum_{g \in \underline{\mathcal{G}}_i} p^{\mathrm{t,disp}}_{g,t} + p^{\mathrm{s,\,t,\,min}}_{i,t} \geq \underline{\mathrm{P}}^{\mathrm{t,\,group}}_{i,t} \quad \forall t \in \mathcal{T}, \forall i \in \underline{\mathcal{I}}^{\mathrm{t}}_t. \tag{7}$$

$$\sum_{g \in \overline{\mathcal{G}}_i} p^{\mathrm{t,disp}}_{g,t} - p^{\mathrm{s,\,t,\,max}}_{i,t} \leq \overline{\mathrm{P}}^{\mathrm{t,\,group}}_{i,t} \quad \forall t \in \mathcal{T}, \forall i \in \overline{\mathcal{I}}^{\mathrm{t}}_t. \tag{8}$$

B. Hydro generation model

In (9), we present the bounds on reservoir volume, turbine discharge and spillage.

$$\begin{aligned} \underline{\mathrm{V}}_h \leq v_{h,t} \leq \overline{\mathrm{V}}_h, 0 \leq q_{h,t} \leq d^{\mathrm{h}}_{h,t} \cdot \overline{\mathrm{Q}}_h, 0 \leq s_{h,t} \leq \overline{\mathrm{S}}_h \\ \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \end{aligned} \tag{9}$$

In (10) and (11), the outflow and inflow, respectively, of each plant are defined, followed by the water balances in (12).

$$v^{\mathrm{out}}_{h,t} - \mathrm{C}^{\mathrm{h}} \cdot (q_{u,t} + s_{u,t}) = 0 \quad \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \tag{10}$$

$$\begin{aligned} v^{\mathrm{in}}_{h,t} - \mathrm{C}^{\mathrm{h}} \cdot \left(\sum_{u \in \mathcal{U}_h} (q_{u,t-\mathrm{d}_{u,h}} + s_{u,t-\mathrm{d}_{u,h}}) \right) = \mathrm{C}^{\mathrm{h}} \cdot \mathrm{I}_{h,t} \\ \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \end{aligned} \tag{11}$$

$$\begin{aligned} v_{h,t} - v_{h,t-1} + v^{\mathrm{out}}_{h,t} - v^{\mathrm{in}}_{h,t} + v^{\mathrm{s,\,out}}_{h,t} - v^{\mathrm{s,\,in}}_{h,t} = 0 \\ \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \end{aligned} \tag{12}$$

The approximation of the hydropower function is defined in the following. First, to each plane of the hydropower function approximation, we associate variable p^{hpf}, as shown in (13).

$$\begin{aligned} p^{\mathrm{hpf}}_{h,j,t} - \mathrm{F}^{\mathrm{vol}}_{h,j} \cdot v_{h,t} - \mathrm{F}^{\mathrm{turb}}_{h,j} \cdot q_{h,t} - \mathrm{F}^{\mathrm{spil}}_{h,j} \cdot s_{h,t} = \mathrm{F}^{\mathrm{const}}_{h,j} \\ \forall h \in \mathcal{H}, \forall j \in \mathcal{F}_h, \forall t \in \mathcal{T}. \end{aligned} \tag{13}$$

For any given triple (q, v, s), only one plane of the approximation must be used. Thus, to each plane, we also associate a binary variable, z^{hpf}. Firstly, constraints (14) ensure that, if a plant is not currently generating power, then no plane should be active. On the other hand, whenever the plant is committed, a single plane must be chosen.

$$\sum_{\forall j \in \mathcal{F}_h} z^{\mathrm{hpf}}_{h,j,t} - d^{\mathrm{h}}_{h,t} = 0 \qquad \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \tag{14}$$

Furthermore, we need to make sure that when a plane is chosen, the plant power output equals the plane's power variable. In contrast, when the plant is not committed, the output must be different from all generation variables associated with the planes. To that end, constraints (15) are added to the model.

$$\begin{aligned} -\overline{\mathrm{P}}^{\mathrm{h}}_h \cdot (1 - z^{\mathrm{hpf}}_{h,j,t}) \leq p^{\mathrm{h}}_{h,t} - p^{\mathrm{hpf}}_{h,j,t} \leq \overline{\mathrm{P}}^{\mathrm{h}}_h \cdot (1 - z^{\mathrm{hpf}}_{h,j,t}) \\ \forall h \in \mathcal{H}, \forall j \in \mathcal{F}_h, \forall t \in \mathcal{T}. \end{aligned} \tag{15}$$

The linear approximation that we use for the hydropower function is concave, thus, we need to make sure that for any point (q, v, s), the plant's output always takes the value from the plane that gives the minimum generation. This is achieved by adding constraints (16) to the model. The hydropower output is further constrained to be within its limits in (17).

$$p^{\mathrm{h}}_{h,t} - p^{\mathrm{hpf}}_{h,j,t} \leq 0 \qquad \forall h \in \mathcal{H}, \forall j \in \mathcal{F}_h, \forall t \in \mathcal{T}. \tag{16}$$

$$\underline{\mathrm{P}}^{\mathrm{h}}_h \cdot d^{\mathrm{h}}_{h,t} \leq p^{\mathrm{h}}_{h,t} \leq \overline{\mathrm{P}}^{\mathrm{h}}_h \cdot d^{\mathrm{h}}_{h,t} \qquad \forall h \in \mathcal{H}, \forall t \in \mathcal{T}. \tag{17}$$

In (18), we present the cost-to-go function, which estimates the future cost as a function of the reservoir-volume decisions at the planning horizon's last period.

$$\alpha - \mathrm{A}^{\mathrm{coeff}}_{j,h} \cdot v_{h,\mathrm{T}} \leq \mathrm{A}^{\mathrm{const}}_j \quad \forall j \in \mathcal{A}. \tag{18}$$

C. Network model

In (19), we present the power balance at each network bus, and (20) bounds the flows on the lines and gives the relation between bus angles and flows (in a DC network model).

$$\begin{aligned} \sum_{g \in \mathcal{G}} p^{\mathrm{t}}_{g,t} + \sum_{h \in \mathcal{H}} p^{\mathrm{h}}_{h,t} + \sum_{l \in \mathcal{L}^{+}_b} f_{l,t} \\ - \sum_{l \in \mathcal{L}^{-}_b} f_{l,t} + p^{\mathrm{s,\,n,\,+}}_{b,t} - p^{\mathrm{s,\,n,\,-}}_{b,t} = \mathrm{L}_{b,t} \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T}. \end{aligned} \tag{19}$$

$$\underline{\mathrm{F}}_{l,t} \leq f_{l,t} = \mathrm{X}_l \cdot \left(\theta_{\mathrm{B}^{+}_l,t} - \theta_{\mathrm{B}^{-}_l,t}\right) \leq \overline{\mathrm{F}}_{l,t} \quad \forall l \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{20}$$

D. Hydrothermal unit commitment

Constraints from the thermal model, (1) - (8), the hydro model, (9) - (18), and the network model, (19) - (20), define the feasible region of the HTUC. The problem is thus fully stated once the objective function is chosen. Since we address a cost-based, centralized HTUC, the objective is to minimize the operation costs. In (21), we give the HTUC used here.

$$\begin{aligned}
\phi^* = \min \quad & \sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} \mathrm{C}_g^{\text{t,cost}}\cdot p_{g,t}^{\text{t}} \\
& + \alpha + \sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{B}} \mathrm{C}^{\text{s,b}}\cdot\left(p_{b,t}^{\text{s, n, +}} + p_{b,t}^{\text{s, n, -}}\right) \\
& + \sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} 10^6\cdot\mathrm{C}^{\text{s,b}}\cdot\left(v_{h,t}^{\text{s, out}} + v_{h,t}^{\text{s, in}}\right) \\
& + \sum_{t\in\mathcal{T}} \mathrm{C}^{\text{s,b}}\cdot\Big(\sum_{i\in\underline{\mathcal{I}}_t^{\text{t}}} p_{i,t}^{\text{s, t, min}} + \sum_{i\in\overline{\mathcal{I}}_t^{\text{t}}} p_{i,t}^{\text{s, t, max}}\Big) \\
\text{s.t.} \quad & (1)\text{–}(8), (9)\text{–}(18), (19)\text{–}(20), \\
& 0 \le y_{g,t}, x_{g,t}, d_{h,t}^{\text{t}} \le 1 \qquad \forall g\in\mathcal{G}, \forall t\in\mathcal{T}, \\
& y_{g,t}, x_{g,t}, d_{g,t}^{\text{t}} \in \{0,1\} \qquad \forall g\in\mathcal{G}, \forall t\in\mathcal{T}, \\
& 0 \le d_{h,t}^{\text{h}} \le 1, d_{h,t}^{\text{h}} \in \{0,1\} \qquad \forall h\in\mathcal{H}, \forall t\in\mathcal{T}.
\end{aligned} \tag{21}$$

The objective function of (21) is composed of three terms: operation costs of thermal units, estimate incurred future cost, and costs of violations. The last term includes violations of power balance at the network buses, and violations of reservoir volumes and the limits on the combined generation of plants.

E. Dual Dynamic Integer Programming

In DDP, the planning horizon of the problem of interest is divided into subhorizons. Then, starting by the first subhorizon, decisions are taken sequentially: firstly for the first subhorizon, which will then serve as input to the second subhorizon, and so on until a complete decision for the entire planning horizon is available. An estimate of subsequent subhorizons optimal cost is iteratively built with dual information from time-coupling constraints and used for evaluating the trade-off between current and future costs. For briefness, we state below the HTUC (21) in a compact form that is suitable for dynamic programming.

$$\phi_{s,j}^* = \min_{x,y} \quad f_s(x_s, y_s) + \nu \tag{22a}$$

$$\text{s.t.} \quad \mathrm{A}\cdot(x_1,...,x_s) + \mathrm{B}\cdot(y_1,...,y_s) \ge \mathrm{C}, \tag{22b}$$

$$(x_r, y_r) - (\hat{\mathbf{x}}_r, \hat{\mathbf{y}}_r) = 0 \quad \forall r\in\{1,...,s-1\}, \tag{22c}$$

$$\begin{aligned}\nu \ge \underline{\phi}_{s+1,i}^* + \pi_{s+1,i}^\top\cdot((\mathbf{x}_{i,1},...,\mathbf{x}_{i,s},\mathbf{y}_{i,1},...,\mathbf{y}_{i,s}) \\ - (x_1,...,x_s,y_1,...,y_s)) \quad \forall i\in\mathcal{I}_s,\end{aligned} \tag{22d}$$

$$y_s \in \{0,1\}^m. \tag{22e}$$

Model (22) is the optimization model of the s subhorizon in iteration j. In this model, the objective function comprises the linear-cost function of the subhorizon costs and the approximation of the subsequent subhorizon optimal cost. In (22), all binary variables of the HTUC are represented by the m-dimensional binary vector y, while x comprises all continuous decisions. All constraints and variables' bounds of the HTUC up to subhorizon s are contained in Eq. (22b), while Eq. (22c) ensure that the decisions taken in previous subhorizons are fixed. An under-estimate of subhorizon $s+1$'s optimal cost as a function of subhorizon s's decision is given by the piece-wise linear model in Eq. (22d), where $i\in\mathcal{I}_s$ is the set of previous iterations in which decisions $(\mathbf{x}_{i,1},...,\mathbf{x}_{i,s},\mathbf{y}_{i,1},...,\mathbf{y}_{i,s})$ have been evaluated in a relaxation of subhorizon $s+1$ and have given optimal costs $\underline{\phi}_{s+1,i}^*$, and the dual information π have been appropriately collected.

For a problem whose planning horizon has been divided into S subhorizons, sequentially solving the subhorizon problems $s\in\{1,...,\mathrm{S}\}$ constitutes a forward step. At the end of this step, a solution and an upper bound to the original problem are available. Then, to improve the approximation of subsequent subhorizons of each subhorizon, a backward step can be performed. In this step, dual information from the coupling constraints are obtained by solving a relaxation of the subhorizon problem. For the relaxation, the values of the dual variables form a valid subgradient and are then used for building a piece-wise linear approximation to the relaxation's optimal cost. Among the relaxations suitable for the backward step, the simplest one for MILPs is the continuous relaxation: accomplished by removing the integrality constraints from the model. Although this relaxation might not give the best bounds, and thus result in weaker approximations, it is satisfactory for the problem at hand and it is used in this work.

F. Cut sharing in dual dynamic programming

In DDP, it is reasonable to expect that as the number of subhorizons increases so does the number of iterations necessary for the algorithm to converge. This mainly happens because more iterations are needed for the approximations of subsequent subhorizons to be good enough. On the other hand, by using fewer subhorizons, the forward and backward problems become larger and so their running times, specially forward's, can become prohibitive. Therefore, one usually seeks a trade-off between smaller subhorizons that are easier to solve and larger subhorizons that give better bounds in fewer iterations. However, for the same class of problems, there is usually no single number of subhorizons that can satisfactorily solve a wide range of the problem's instances. For time-sensitive problems that need to be solved regularly, as many practical problems do, this can be a obstacle to applications of DDP since one would need to test several numbers of subhorizons every time the problem is to be solved.

One way to partially overcome the aforementioned difficulties is to share optimality cuts among different partitions. For example, take a problem with a 48-period planning horizon, and consider the partitions shown in Fig. 1. For convenience and for later use, each of the partitions are assigned to processes. For all partitions in Fig. 1, period 24 is the last period of the subhorizon that precedes period 25's subhorizon. Thus, for all five processes, the recourse function of the subhorizon whose last period is the 24th are identical: ultimately, they all represent the costs of periods 25 to 48. Consequently, in DDP, the approximations of the recourse functions for period 24 that one would get by considering any of the partitions shown in Fig. 1 can be shared among all processes. Evidently, adding cuts from a coarser partition, as 48 subhorizons, to a finer partition as, two subhorizons, might not bring many benefits. In contrast, the reverse can significantly improve the bounds provided by coarser partitions.

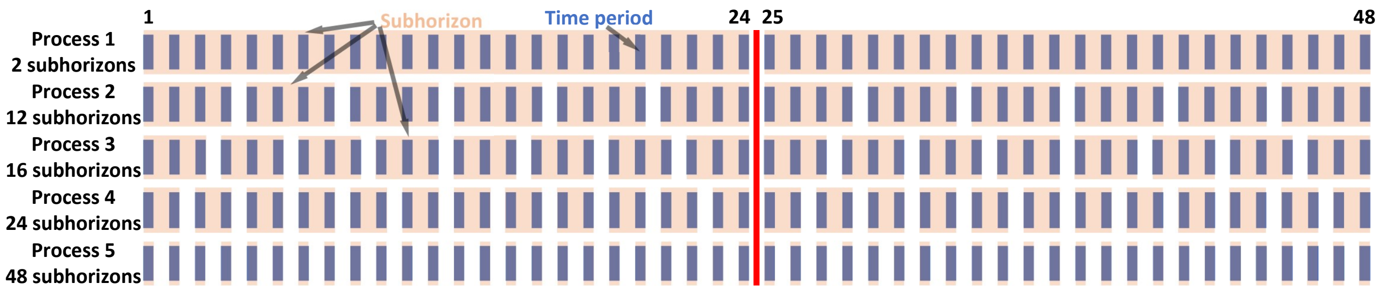


Fig. 1: Example of cut sharing for a problem with 48 periods.

In a sequential setting, the sharing scheme discussed above can be implemented as follows. Choose a coarse partition to be used in the forward and backward steps. Additionally, choose a finer partition that can share optimality cuts with the first one — this partition will be only used in the backward step. Proceed with the forward process as usual. Once a primal solution is available, evaluate it in the backward step with the finer partition and add the derived cuts to the appropriate subhorizons of the coarser one.

Although simple and significantly beneficial, to the best of the authors' knowledge, this strategy is not found in the literature. Despite its potential benefits, cut sharing alone in the sequential setting presented above might not be able to meet the performance required for application in practice since the convergence would still be dependent upon a single partition in the forward step. Thus, a natural next step is to employ parallelism both in the forward and backward step to exploit the distinct characteristics of different time partitions.

III. Parallel dual dynamic integer programming

Our parallel DDiP consists of simultaneously solving multiple partitions of the same problem and sharing among them dual information. The idea behind this strategy is that, for the forward step, coarser partitions give fast, but likely poorer, primal solutions. These possibly poor primal solutions can be evaluated in a backward step with any partition generally faster than a forward step can be performed. Thus, the dual information derived from these poor solutions can be used even in the finer partitions, hence quickly improving the approximations of the recourse functions.

Moreover, in this strategy, multiple processes with the same partition can be used in the backward step to simultaneously evaluate different primal solutions. Therefore possibly improving the approximation of the recourse functions to a point that a classical sequential strategy would take several iterations to reach. For instance, suppose that processes 2, 3, 4 and 5 from Fig. 1 are all running a forward step. Thus, once all of them are finished, there will be 4 primal solutions, likely different from each other, that can be evaluated in the backward step. If four processes with identical two-subhorizon partitions are available, each of them can take one of the primal solutions. Then, once the problems associated with the 2nd subhorizon are solved, dual information concerning periods 25 to 48 from four different solutions will be available to processes 2, 3, 4 and 5.

In our implementation, we use the figure of a general coordinator: a process responsible for receiving, and distributing information from the worker processes, while also checking the convergence of the algorithm. The optimization of the subhorizon problems are carried by what we call forward and backward workers. The first solely perform forward steps, sending primal solutions and upper bound information to the general coordinator, and receiving back dual information; the last solve only the backward subhorizon problems. For clarity and conciseness, we present in Alg. 1 a simplified, high-level version of the role of the general coordinator.

Algorithm 1: General coordinator.

```
Input: $tol > 0$
Output: $LB$ and $UB$
1  $LB \leftarrow 0$, $UB \leftarrow \infty$;
2  while $((UB - LB)/UB > tol)$ do
3      $(s, newLB, newUB) \leftarrow receiveSolution()$;
4      if $s$ is a primal solution then
5          $sendToBackward(s)$;
6          $updateLB(newLB)$;
7          $updateUB(newUB)$;
8      else if $s$ is a dual solution then
9          $sendToBackward(s)$;
10         $sendToForward(s)$;
11         $updateLB(newLB)$ ;
```

In Alg. 1, we omit all stopping criteria but the relative gap. In Step 3 of Alg. 1, the general coordinator waits for a solution from either a forward or a backward worker. If a forward solution is available, then it is sent to a backward worker. In practice, a backward worker might not be readily available, thus the primal solution must be temporarily stored — we also omit this from Alg. 1 for simplicity. Similarly, note that lower bound information from a forward worker is available as soon as the first subhorizon is solved. While in the actual implementation of the algorithm, we exploit this, we leave it out of Alg. 1. In contrast, when a dual solution is received from a backward worker, it is sent to forward and also backward workers: to forward workers as long as the partitions are the same; and to backward workers as long as there are matches between the receiving and sending workers, and the partition from the sending worker has fewer subhorizons. Note that we have included in Step 11 of Alg. 1 the possibility of updating the lower bound from the information of a backward worker. This updating procedure is possible as long as the backward worker solves its first subhorizon problem. Now, in Alg. 2, we present the steps taken by the forward and backward workers, respectively.

Algorithm 2: Forward and backward workers.

```
1  Procedure $ForwardWorker(tol)$:
2      $LB \leftarrow 0$, $UB \leftarrow \infty$;
3      while $((UB - LB)/UB > tol)$ do
4          $x \leftarrow forwardStep()$;
5          $sendPrimalSol(x, LB, UB)$;
6          $(y, newLB, newUB) \leftarrow receiveDualSol()$;
7          $updateLB(newLB)$;
8          $updateUB(newUB)$;
9  Procedure $BackwardWorker(tol)$:
10     $LB \leftarrow 0$, $UB \leftarrow \infty$;
11     while $((UB - LB)/UB > tol)$ do
12         $(x, newLB, newUB) \leftarrow receivePrimalSol()$;
13         $updateLB(newLB)$;
14         $updateUB(newUB)$;
15         $y \leftarrow backwardStep()$;
16         $sendDualSol(y, LB, UB)$;
```

An important characteristic of the proposed algorithm is its asynchronous nature. Since the order of the cuts added to forward and backward workers depend on how fast the associated problems are solved, the dual and primal solutions yielded by backward and forward workers might change each time the algorithm is run. Therefore, the total running times, the bounds, and the best primal solution found may also change with each new run.

IV. Computational experiments

To evaluate the computational performance of the proposed algorithm, we use 20 test-cases with 48 half-hour periods derived from real problems of the Brazilian power system. The power system has 329 thermal units, 161 hydro plants, 7,475 buses, and 10,702 transmission lines. The cases, identified as Case 1 to Case 20, were chosen to reflect vast operation conditions: low loads in winter and high loads in summer; periods of low and high inflows; as well as different initial conditions. To showcase the differences among the cases, we present in Fig. 2 the net loads for the 20 cases.

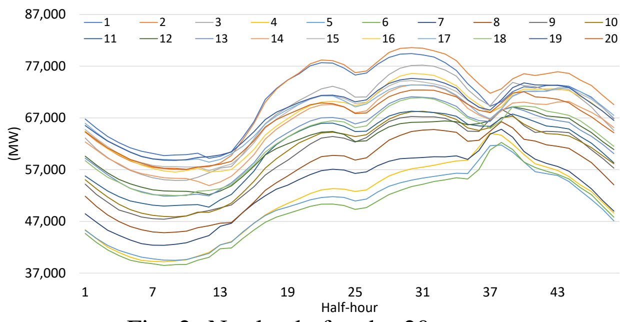


Fig. 2: Net loads for the 20 cases.

Table I shows the number of constraints and variables of the HTUC tackled in this work. The numbers are taken from Case 1; despite small differences in dimension due to variations in the hydropower functions caused by distinct initial conditions, all cases have roughly the same dimensions.

TABLE I: Dimensions of the optimization problem of Case 1 with a single subhorizon.

Constraints	Continuous variables	Binary variables
679,417	616,762	74,880

Although countless variations of configurations of forward and backward workers in the parallel DDiP are possible, we perform our experiments with a single one: five forward processes, and ten backward processes. All processes have uniform distributions of periods, i.e., all their subhorizons have the same number of periods. The forwards have 6, 8, 12, 16 and 24 subhorizons. For the backwards, we dedicate five of them to solving two-subhorizon partitions. One of these five workers solves only the first subhorizon — it is assigned to obtaining lower bounds. In contrast, the remaining four two-subhorizon backward workers only solve the 2nd subhorizon in an attempt to continuously provide quality optimality cuts for all other backward processes. In addition to these five two-subhorizon backward workers, we have five others with the same partitions as the forwards': 6, 8, 12, 16, and 24. Due to the asynchronous nature of the algorithm, we run each case with the parallel strategy five times.

We compare our strategy with the classical, sequential DDiP with 1, 2, 3, 4, 6, 8, 12, 16, 24, and 48 subhorizons. Evidently, using a single subhorizon boils down to solving the problem without DDiP. For this particular case, the time limit is set to 24 h, while for all others, including the parallel strategy, the time limit is set to 2 h. For all sequential experiments, Gurobi is free to use all threads of the machine, whereas in the parallel strategy, we allow Gurobi to use at most two threads in each process. Moreover, a tolerance of 0.1% is set to the relative gap for all cases. Except for the experiments with a single subhorizon, the forward problems have a tolerance for the relative gap of 0.01%.

All experiments are conducted on a workstation running Ubuntu 20.04 with 128 GB of RAM and 2 Intel Xeon E5-2660 v3 @ 2.60 GHz, totaling 20 physical cores (40 threads). The optimization solver used in all experiments is Gurobi 9.5.0 [14], which is set to the same configuration in all experiments; the programming language is Python, and the interpreter is PyPy 7.3.5. Communication between processes is performed through Open MPI 4.1.1, with mpi4py [15] serving as a interface between Python and Open MPI's libraries.

V. Results

The first results are shown in Table II for the approach with no temporal decomposition.

TABLE II: Results with no temporal decomposition. "0" is used when a lower bound cannot be found; and "-" when no primal solution is found.

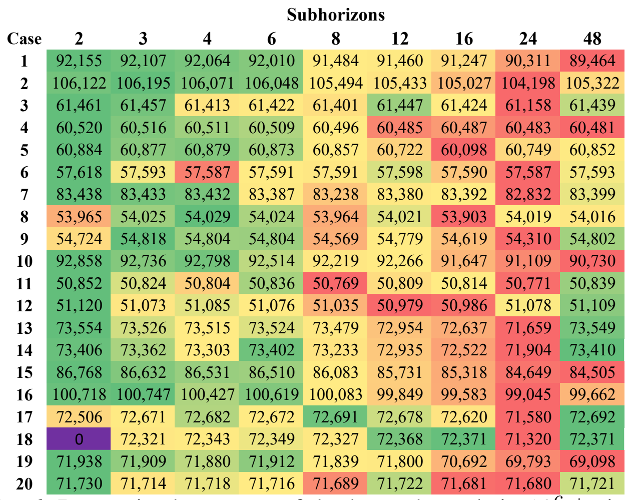
Case	Lower bound ($)	Upper bound ($)	Gap (%)	Time (hour)
1	92,182,198,792.44	92,182,249,661.40	$5.5 \cdot 10^{-5}$	12
2	106,284,041,140.70	-	$1.0 \cdot 10^{2}$	24
3	0.00	-	$1.0 \cdot 10^{2}$	24
4	60,524,854,888.94	60,524,930,909.12	$1.3 \cdot 10^{-4}$	15
5	60,886,178,298.09	60,886,192,199.84	$2.3 \cdot 10^{-5}$	16
6	57,622,716,467.41	57,622,748,250.16	$5.5 \cdot 10^{-5}$	15
7	83,441,391,572.63	83,441,414,004.50	$2.7 \cdot 10^{-5}$	16
8	54,054,723,957.43	54,335,471,255.07	$5.2 \cdot 10^{-1}$	24
9	54,828,212,808.82	-	$1.0 \cdot 10^{2}$	24
10	92,876,126,005.14	92,876,196,169.10	$7.6 \cdot 10^{-5}$	14
11	50,871,544,591.47	-	$1.0 \cdot 10^{2}$	24
12	51,141,075,890.34	51,141,095,404.99	$3.8 \cdot 10^{-5}$	19
13	73,602,153,436.05	-	$1.0 \cdot 10^{2}$	24
14	73,474,584,291.03	73,474,622,815.43	$5.2 \cdot 10^{-5}$	22
15	86,881,571,786.03	92,301,323,915.46	$5.9 \cdot 10^{0}$	24
16	100,820,024,189.07	100,820,121,642.89	$9.7 \cdot 10^{-5}$	13
17	72,741,130,711.90	72,741,189,124.99	$8.0 \cdot 10^{-5}$	15
18	72,418,781,478.38	72,418,849,408.51	$9.4 \cdot 10^{-5}$	22
19	72,020,032,424.69	-	$1.0 \cdot 10^{2}$	24
20	71,773,276,086.21	-	$1.0 \cdot 10^{2}$	24

From Tab. II, we see that, even after 24 h, Gurobi cannot to find a solution to 7 of the 20 cases. Convergence is reached for 11 of the cases with near-zero gaps, but at the cost of more than 12 h of running time. For two other cases, Case 8 and Case 15, Gurobi finds a solution but is not able to prove its quality. By analysing the progress of Gurobi's optimization algorithm, we see that the lower bound stabilizes early on in the optimization process: it hardly changes after the root relaxation is solved. Thus, apart from Case 3, for which Gurobi is unable to solve the root node in 24 h, the lower bounds from Tab. II can be treated as a good estimate of the true optimal value, and it will be used as a reference to the quality of the bounds found by the DDP algorithms.

Following the results of Tab. II, we now show in Fig. 3 and Fig. 4 the results for what we call the sequential strategy. In addition to serving as benchmark, these results give us the necessary motivation to devise a parallel strategy.

The results from Fig. 3 and Fig. 4 show that no single partition is capable of solving all cases to 0.1%. Surprisingly, the most successful partition is the one with 48 subhorizons with a total of 14 convergences out of 20. However, the non-convergent cases with 48 subhorizons, namely, cases 1, 2, 10, 15, 16, and 19, all finished the two-hour allowed running time with a relative gap greater than 1%. Moreover, for two cases, 2 and 19, none of the partitions is able to achieve a 0.1% gap, further demonstrating the difficulty of the problem at hand. Considering only the convergent experiments, the partition that delivers the best average running time is the one with six subhorizons with an average of 38 min.

If we divide Fig. 3 and Fig. 4 vertically into experiments with at most six subhorizons and experiments with at least eight subhorizons, we see that most convergent experiments are in the left side — 38 of the total 68 convergent experiments. To better understand why most convergent cases have at most six subhorizons, we can analyse the upper and lower bounds yielded by different partitions. As absolute differences can cloud the analysis, we rely on relative metrics. For both bounds, the variation metric used is $(B^{\max} - B^{\min})/B^{\max}$, where B can be either the lower bound or the upper bound, and $B^{\max}$ and $B^{\min}$ are, respectively, the maximum and minimum bound for a particular case. Interestingly, the upper bounds do not vary significantly over the number of subhorizons. In fact, the case with most variation, apart from case 18, for which no solution was found, is Case 10 with 0.11%; the case with least variation is Case 20 with 0.03%. Therefore, the large relative gaps that we see in Fig. 3 for the non-convergent experiments must be mainly do to poor lower bounds. In Fig. 6, we present the lower bounds for each case and partition.

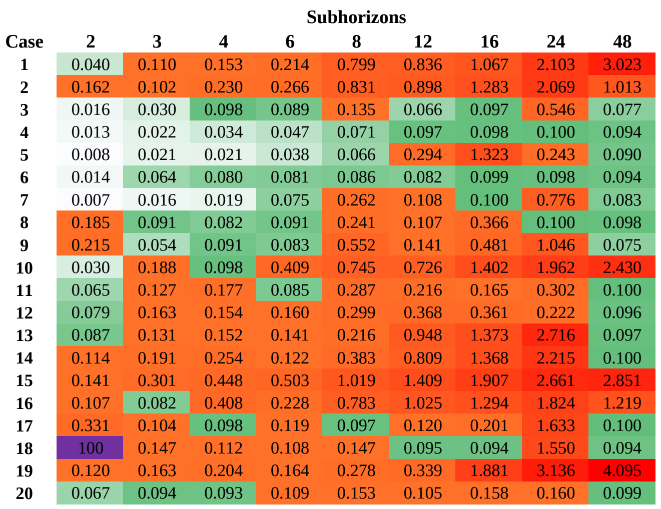


	Subhorizons								
Case	2	3	4	6	8	12	16	24	48
1	92,155	92,107	92,064	92,010	91,484	91,460	91,247	90,311	89,464
2	106,122	106,195	106,071	106,048	105,494	105,433	105,027	104,198	105,322
3	61,461	61,457	61,413	61,422	61,401	61,447	61,424	61,158	61,439
4	60,520	60,516	60,511	60,509	60,496	60,485	60,487	60,483	60,481
5	60,884	60,877	60,879	60,873	60,857	60,722	60,098	60,749	60,852
6	57,618	57,593	57,587	57,591	57,591	57,598	57,590	57,587	57,593
7	83,438	83,433	83,432	83,387	83,238	83,380	83,392	82,832	83,399
8	53,965	54,025	54,029	54,024	53,964	54,021	53,903	54,019	54,016
9	54,724	54,818	54,804	54,804	54,569	54,779	54,619	54,310	54,802
10	92,858	92,736	92,798	92,514	92,219	92,266	91,647	91,109	90,730
11	50,852	50,824	50,804	50,836	50,769	50,809	50,814	50,771	50,839
12	51,120	51,073	51,085	51,076	51,035	50,979	50,986	51,078	51,109
13	73,554	73,526	73,515	73,524	73,479	72,954	72,637	71,659	73,549
14	73,406	73,362	73,303	73,402	73,233	72,935	72,522	71,904	73,410
15	86,768	86,632	86,531	86,510	86,083	85,731	85,318	84,649	84,505
16	100,718	100,747	100,427	100,619	100,083	99,849	99,583	99,045	99,662
17	72,506	72,671	72,682	72,672	72,691	72,678	72,620	71,580	72,692
18	0	72,321	72,343	72,349	72,327	72,368	72,371	71,320	72,371
19	71,938	71,909	71,880	71,912	71,839	71,800	70,692	69,793	69,098
20	71,730	71,714	71,718	71,716	71,689	71,722	71,681	71,680	71,721

Fig. 6: Row-wise heatmaps of the lower bounds in 10^6 \$ given by the sequential DDiP. Greener cells indicate better lower bounds, while redder ones show the worse. Conveniently, Case 18 with two subhorizons is shown in purple.

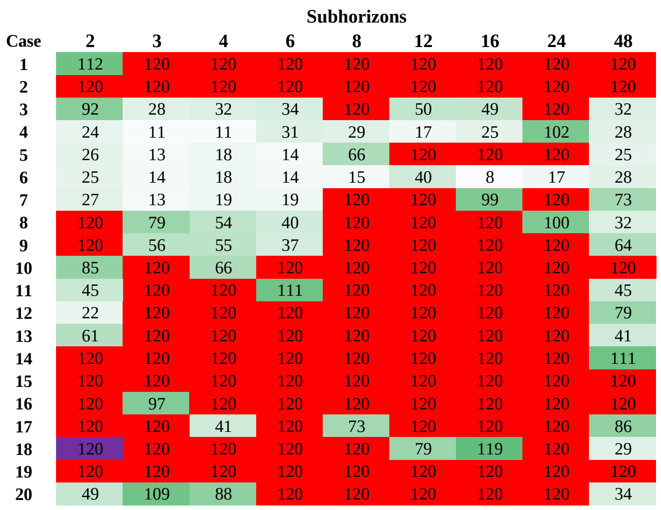
Fig. 6 shows that partitions with at most 6 subhorizons provide the best lower bounds — the two exceptions are cases 17 and 18. Unlike the upper bounds, the relative variation for the lower bound can be as much as 3.94% for Case 19, with an average over all cases, except Case 18, of 1.35%. In fact, we can assess the quality of the upper bounds given by the sequential strategy by comparing the worst of them case-wise (row-wise in Fig. 6) with the lower bounds from Tab. II. For such analysis, we do not consider cases 3 and 18, and we use the difference between the worst upper bound from the sequential strategy and the lower bound from Tab. II w.r.t. lower bound from Tab. II. With this metric, the worst upper bound is Case 10's whose relative distance is 0.12%. Following the same reasoning for the lower bound, we find that the worst lower bounds from Fig. 6 are significantly more distant from those of Tab. II. The worst case is 19 whose relative distance is about 4% (due to the lower bound given by the partition with 48 subhorizons).

The tighter lower bounds produced by the partitions with at most six subhorizons are partly due to their approximations of the recourse functions. Assuming that to be true, we can then evaluate the computational costs, in terms of running times, of such better approximations. In Tab. III, we give the total running times of the backward problems. As we can see in Tab. III, the running times do not vary significantly



	Subhorizons								
Case	**2**	**3**	**4**	**6**	**8**	**12**	**16**	**24**	**48**
1	0.040	0.110	0.153	0.214	0.799	0.836	1.067	2.103	3.023
2	0.162	0.102	0.230	0.266	0.831	0.898	1.283	2.069	1.013
3	0.016	0.030	0.098	0.089	0.135	0.066	0.097	0.546	0.077
4	0.013	0.022	0.034	0.047	0.071	0.097	0.098	0.100	0.094
5	0.008	0.021	0.021	0.038	0.066	0.294	1.323	0.243	0.090
6	0.014	0.064	0.080	0.081	0.086	0.082	0.099	0.098	0.094
7	0.007	0.016	0.019	0.075	0.262	0.108	0.100	0.776	0.083
8	0.185	0.091	0.082	0.091	0.241	0.107	0.366	0.100	0.098
9	0.215	0.054	0.091	0.083	0.552	0.141	0.481	1.046	0.075
10	0.030	0.188	0.098	0.409	0.745	0.726	1.402	1.962	2.430
11	0.065	0.127	0.177	0.085	0.287	0.216	0.165	0.302	0.100
12	0.079	0.163	0.154	0.160	0.299	0.368	0.361	0.222	0.096
13	0.087	0.131	0.152	0.141	0.216	0.948	1.373	2.716	0.097
14	0.114	0.191	0.254	0.122	0.383	0.809	1.368	2.215	0.100
15	0.141	0.301	0.448	0.503	1.019	1.409	1.907	2.661	2.851
16	0.107	0.082	0.408	0.228	0.783	1.025	1.294	1.824	1.219
17	0.331	0.104	0.098	0.119	0.097	0.120	0.201	1.633	0.100
18	100	0.147	0.112	0.108	0.147	0.095	0.094	1.550	0.094
19	0.120	0.163	0.204	0.164	0.278	0.339	1.881	3.136	4.095
20	0.067	0.094	0.093	0.109	0.153	0.105	0.158	0.160	0.099

Fig. 3: Relative gaps (%).



	Subhorizons								
Case	**2**	**3**	**4**	**6**	**8**	**12**	**16**	**24**	**48**
1	112	120	120	120	120	120	120	120	120
2	120	120	120	120	120	120	120	120	120
3	92	28	32	34	120	50	49	120	32
4	24	11	11	31	29	17	25	102	28
5	26	13	18	14	66	120	120	120	25
6	25	14	18	14	15	40	8	17	28
7	27	13	19	19	120	120	99	120	73
8	120	79	54	40	120	120	120	100	32
9	120	56	55	37	120	120	120	120	64
10	85	120	66	120	120	120	120	120	120
11	45	120	120	111	120	120	120	120	45
12	22	120	120	120	120	120	120	120	79
13	61	120	120	120	120	120	120	120	41
14	120	120	120	120	120	120	120	120	111
15	120	120	120	120	120	120	120	120	120
16	120	97	120	120	120	120	120	120	120
17	120	120	41	120	73	120	120	120	86
18	120	120	120	120	120	79	119	120	29
19	120	120	120	120	120	120	120	120	120
20	49	109	88	120	120	120	120	120	34

Fig. 4: Total running time (min).

Fig. 5: Heatmaps of the sequential DDiP. For convergent cases, data is giving in a green scale: the closer to white, the better; for non-convergent cases, the scale is red: the redder, the worse. Cases for which no solution was obtained are in purple.

in absolute terms over the number of subhorizons. Therefore, obtaining the (hopefully) better subgradients from partitions with fewer subhorizons takes roughly the same time as the poorer subgradients from partitions with more subhorizons.

TABLE III: Average backward time in seconds for each case and partition. (The first subhorizon problem is not solved).

	Number of subhorizons								
Case	2	3	4	6	8	12	16	24	48
1	39	40	40	36	34	31	29	29	32
2	44	44	42	40	36	32	30	30	34
3	55	99	53	41	38	32	30	29	32
4	60	50	47	41	37	34	30	30	31
5	50	48	44	42	37	33	30	30	32
6	47	48	44	40	37	32	32	29	32
7	56	49	44	40	37	32	31	30	32
8	41	48	45	41	38	34	31	30	32
9	39	50	45	42	36	34	31	29	32
10	39	41	39	37	34	31	29	29	35
11	64	52	47	43	38	33	31	30	33
12	54	45	43	39	36	33	30	30	33
13	59	50	45	42	38	32	31	29	33
14	54	50	46	42	37	32	31	30	35
15	40	43	39	38	37	31	29	29	34
16	40	41	38	37	35	31	29	29	34
17	54	52	46	40	36	32	29	30	33
18	-	52	46	39	36	32	29	29	32
19	56	51	45	42	37	33	30	29	32
20	51	52	45	41	40	35	30	30	32
Average	50	50	44	40	37	33	30	29	33

To empirically assess the benefits of using subgradients from partitions with fewer subhorizons, we use as an example Case 1. For this case, we take the first iterates obtained by partitions with 12, 16, 24 and 48 subhorizons. These iterates are then evaluated in the second subhorizon of a two-subhorizon partition — giving then four subgradients, one for each iterate. Then, in the backward step of the first iteration for partitions 12, 16, 24 and 48, we first appropriately add the newly obtained linear inequalities from the 2-subhorizon partition to the subhorizon whose last period immediately precedes period 25th. Afterwards, the backward step is carried out as usual. After this step is finished, the second iteration begins. Once the first subhorizon is solved in this iteration, we have a new lower bound; once the last one is finished, a new solution is available. Then, we analyse the benefits of using the 2-subhorizon cuts by comparing the lower bound and the cost of the solution in the second iteration with and without these additional cuts. Tab. IV summarizes our findings.

TABLE IV: Comparison of lower bound and solution cost with and without additional cuts from a 2-subhorizon partition for Case 1. The improvements are measured as the ratio of the difference between the metric with no additional cuts and that with additional cuts over the metric with no additional cuts.

Subhorizons/ Additional cuts	Lower bound/Current solution (10^6 \$)	Lower bound/Solution cost Improvement (%)
12 / no	90,826 / 92,530	- / -
12 / yes	91,231 / 92,405	0.4 / 0.1
16 / no	90,687 / 92,527	- / -
16 / yes	91,298 / 92,373	0.7 / 0.2
24 / no	1.537 / 92,691	- / -
24 / yes	796 / 92,377	51,679.1 / 0.3
48 / no	0.799 / 93,018	- / -
48 / yes	0.799 / 92,759	0.0 / 0.3

Tab. IV shows that adding the cuts from the 2-subhorizon partition improves both the lower bounds and the solution costs in the subsequent iteration. Particularly, we can see a significant increase in the lower bound for the partition with 24 subhorizons. On the other hand, the partition with 48 subhorizons remains with the same lower bound, but sees a considerable decrease in its solution cost.

In contrast to the backward step, the forward step's total time varies considerably according to the number of subhorizons chosen. In Tab. V, we show the average of the forward running times for the 20 cases.

TABLE V: Average forward time in seconds.

	Number of subhorizons								
Case	2	3	4	6	8	12	16	24	48
1	2,131	723	599	408	277	192	143	102	71
2	2,180	887	603	437	382	223	144	123	75
3	2,609	726	580	368	266	169	127	101	74
4	1,250	591	581	329	252	172	121	100	74
5	1,414	707	494	365	293	179	151	110	73
6	1,367	751	483	376	250	169	126	87	72
7	1,485	705	504	325	307	171	130	100	74
8	1,635	733	488	361	298	182	132	101	72
9	1,743	776	611	400	283	186	140	110	74
10	1,211	701	522	349	291	178	140	107	77
11	2,095	1,045	494	373	261	175	121	86	70
12	1,192	1,120	502	331	240	167	121	87	71
13	3,415	886	670	406	251	184	152	124	71
14	1,428	908	608	351	249	193	132	113	73
15	1,155	730	573	304	258	198	171	105	74
16	1,427	785	777	341	301	198	159	118	76
17	2,370	1,022	550	372	275	184	138	107	77
18	-	1,974	684	363	284	174	141	118	73
19	1,573	1,031	645	422	287	180	136	104	75
20	1,354	1,577	705	324	280	165	138	93	72
Av.	1,739	919	584	365	279	182	138	105	73

As one can see in Tab. V, performing the forward step for a partition with two subhorizons takes nearly 23 times the time taken to do the same for a partition with 48 subhorizons. Comparing the times from Tab. V and Tab. III, we attribute the substantial increase in the running times to the binary variables of the problems in the forward step. More importantly, however, is to note that, although partitions with fewer subhorizons are more likely to give better solutions, after two hours of running time, the upper bounds provided by all partitions are similar. Therefore, it is possible to find quality solutions with partitions of many subhorizons that are easier to solve.

With the 20 cases chosen in this work, we have shown the complexity of partitioning the planning horizon in DDP. Furthermore, we have presented empiric results that indicate that, generally, partitions with fewer subhorizons give better lower bounds, while comparable upper bounds can be obtained with both few and many subhorizons. Now, we combine the tight bounds given by partitions with few subhorizons with the fast upper bounds from partitions with many subhorizons in a parallel DDiP strategy. Table VI shows the relative gaps and times for this approach.

Different from the sequential method, we see in Table VI that the parallel strategy is able to solve all 20 cases, in all five trials of each case, to a 0.1% well before the 2-h time limit is reached. Considering the best times of the sequential strategy for each case and the averages of the parallel one, the average speed-up of the parallel strategy over the sequential method is over four. On the other hand, using the parallel strategy's worst times instead gives an average (worst) speed-up of about 3.6. For only one case the worst parallel time is greater than the best sequential time: Case 10. And, for this particular case, only one of its five trials fails to be better than all sequential times. Comparing the bounds given by the parallel strategy with those from Tab. II shows that both lower and upper bounds are close to the optimal: ignoring Case 3, the farthest lower bound and the farthest upper bound are both only 0.08% of the optimal value.

As shown in the problem formulation, we choose not to use feasibility cuts but slack variables instead. A possible shortcoming of this approach is that the slack variables might not be small enough. In Tab. VII, we show the largest values of the slack variables in Eq. (12), Eq. (7), and Eq. (8), for each case over their five trials. The volume slack variables take negligible values, whereas the thermal slacks, those of Eq. (7) and Eq. (8), take values greater than zero for 25% of the cases. The largest violation happens for Case 16 with 74.4 MW. While this value is not negligible, but we argue that it could be further reduced by increasing the penalty factor. More importantly, however, it is not sufficiently significant to affect the quality of the solution — note that the minimum load for this system is about 37 GW.

TABLE VII: Largest values of the slack variables in the results of the parallel strategy. In this table, ”v” stands for volume and ”th” stands for thermal.

Case	V. slacks (hm^3)	Th. slacks (MW)	Case	V. slacks (hm^3)	Th. slacks (MW)
1	$3.81 \cdot 10^{-7}$	$3.79 \cdot 10^{-10}$	11	$4.63 \cdot 10^{-7}$	$1.11 \cdot 10^{-13}$
2	$5.24 \cdot 10^{-7}$	$7.06 \cdot 10^{-9}$	12	$7.07 \cdot 10^{-9}$	$4.44 \cdot 10^{-14}$
3	$1.19 \cdot 10^{-9}$	$4.44 \cdot 10^{-14}$	13	$2.04 \cdot 10^{-8}$	72.6
4	$1.75 \cdot 10^{-7}$	$1.22 \cdot 10^{-13}$	14	$1.55 \cdot 10^{-7}$	$4.44 \cdot 10^{-14}$
5	$2.89 \cdot 10^{-7}$	$1.22 \cdot 10^{-13}$	15	$1.86 \cdot 10^{-7}$	6
6	$9.69 \cdot 10^{-8}$	$3.11 \cdot 10^{-13}$	16	$3.08 \cdot 10^{-9}$	74.4
7	$1.99 \cdot 10^{-7}$	59.7	17	$2.61 \cdot 10^{-11}$	$4.44 \cdot 10^{-14}$
8	$5.67 \cdot 10^{-12}$	60	18	$7.38 \cdot 10^{-8}$	$5.55 \cdot 10^{-14}$
9	$2.12 \cdot 10^{-7}$	$1.83 \cdot 10^{-13}$	19	$6.59 \cdot 10^{-8}$	$1.11 \cdot 10^{-14}$
10	$5.79 \cdot 10^{-8}$	$6.33 \cdot 10^{-5}$	20	$3.02 \cdot 10^{-11}$	$2.22 \cdot 10^{-13}$

To exemplify the results from the HTUC, we present in Fig. 7 the total hydro and thermal generations for Case 16 over its 48 periods. The generation profile are typical for this system: the hydro generation follows the load changes, while the thermal generation keeps a flatter profile.

VI. Conclusion

We have shown how the strengths of different temporal decomposition can be used together to consistently deliver strong results through a novel parallel dual dynamic method. We have evaluated our method on 20 cases of a large-scale hydrothermal unit-commitment problem that is provably challenging to the state-of-the-art, off-the-shelf optimization solver. Our results show that our strategy performs satisfactorily for all test cases providing significant speed-ups in addition to solving problems that cannot be solved by the benchmark strategy. Opportunities for future work within the proposed method may include further improving the hydrothermal unit-commitment model and extending the methodology to include accelerating techniques.

References

[1] M. V. Pereira and L. M. Pinto, “Multi-stage stochastic optimization applied to energy planning,” *Mathematical programming*, vol. 52, no. 1, pp. 359–375, 1991.

[2] J. Zou, S. Ahmed, and X. A. Sun, “Stochastic dual dynamic integer programming,” *Mathematical Programming*, vol. 175, no. 1-2, pp. 461–502, Mar. 2018. [Online]. Available: https://doi.org/10.1007/s10107-018-1249-5

[3] ——, “Multistage stochastic unit commitment using stochastic dual dynamic integer programming,” *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 1814–1823, 2019.

[4] M. N. Hjelmeland, J. Zou, A. Helseth, and S. Ahmed, “Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming,” *IEEE Transactions on Sustainable Energy*, vol. 10, no. 1, pp. 481–490, 2019.

TABLE VI: Main results of the parallel DDiP strategy. Column "Best time of seq. strategy (min)" is the row-wise minimum time in minutes, from Fig. 4, of the sequential strategy. The speed-up column gives a pair of speed-up metrics. The first is the ratio of the best sequential time over average running time of the parallel strategy, while the second is what we call " worst speed-up", which is that same ratio but with the worst parallel time.

Case	Gap (%) (min, average, max)	Time (min) (min, average, max)	Best time of seq. strategy (min)	Speed-up	Lower bound (10^6 \$) (min, average, max)
1	(0.03, **0.045**, 0.067)	(33, **45**, 53)	112	(2.5, 2.1)	(92,157.13, **92,163.61**, 92,169.88)
2	(0.043, **0.082**, 0.1)	(29, **39**, 48)	120	(3.1, 2.5)	(106,196.52, **106,231.51**, 106,267.96)
3	(0.087, **0.088**, 0.09)	(7, **7**, 7)	28	(4.0, 4.0)	(61,428.39, **61,428.39**, 61,428.39)
4	(0.09, **0.093**, 0.094)	(4, **5**, 5)	11	(2.2, 2.2)	(60,484.98, **60,484.98**, 60,484.98)
5	(0.021, **0.021**, 0.021)	(8, **8**, 8)	13	(1.7, 1.7)	(60,882.85, **60,882.85**, 60,882.85)
6	(0.059, **0.059**, 0.059)	(5, **5**, 5)	8	(1.6, 1.6)	(57,610.89, **57,610.89**, 57,610.89)
7	(0.024, **0.024**, 0.024)	(7, **7**, 7)	13	(1.9, 1.9)	(83,438.54, **83,438.54**, 83,438.54)
8	(0.091, **0.091**, 0.091)	(8, **8**, 8)	32	(4.0, 4.0)	(54,044.31, **54,044.31**, 54,044.31)
9	(0.08, **0.084**, 0.09)	(12, **13**, 14)	37	(2.9, 2.7)	(54,820.31, **54,822.91**, 54,823.91)
10	(0.019, **0.05**, 0.067)	(20, **40**, 98)	66	(1.6, 0.7)	(92,857.48, **92,866.68**, 92,869.96)
11	(0.088, **0.092**, 0.094)	(10, **13**, 15)	45	(3.4, 3.0)	(50,862.33, **50,864.40**, 50,866.15)
12	(0.077, **0.087**, 0.099)	(13, **14**, 18)	22	(1.6, 1.2)	(51,129.15, **51,129.88**, 51,130.46)
13	(0.082, **0.092**, 0.1)	(10, **12**, 15)	41	(3.4, 2.7)	(73,569.54, **73,579.36**, 73,589.93)
14	(0.092, **0.094**, 0.097)	(12, **14**, 18)	111	(7.9, 6.2)	(73,445.89, **73,451.41**, 73,453.44)
15	(0.062, **0.089**, 0.099)	(13, **16**, 21)	120	(7.5, 5.7)	(86,838.36, **86,846.23**, 86,859.78)
16	(0.07, **0.086**, 0.095)	(9, **14**, 18)	97	(6.9, 5.4)	(100,763.78, **100,782.69**, 100,804.93)
17	(0.052, **0.079**, 0.097)	(8, **8**, 8)	41	(5.1, 5.1)	(72,687.94, **72,702.46**, 72,724.34)
18	(0.052, **0.057**, 0.06)	(7, **8**, 9)	29	(3.7, 3.2)	(72,390.47, **72,391.80**, 72,393.73)
19	(0.054, **0.063**, 0.068)	(12, **13**, 14)	120	(9.2, 8.6)	(71,985.87, **71,989.63**, 71,996.36)
20	(0.09, **0.09**, 0.09)	(5, **5**, 5)	34	(6.9, 6.9)	(71,733.48, **71,733.48**, 71,733.48)

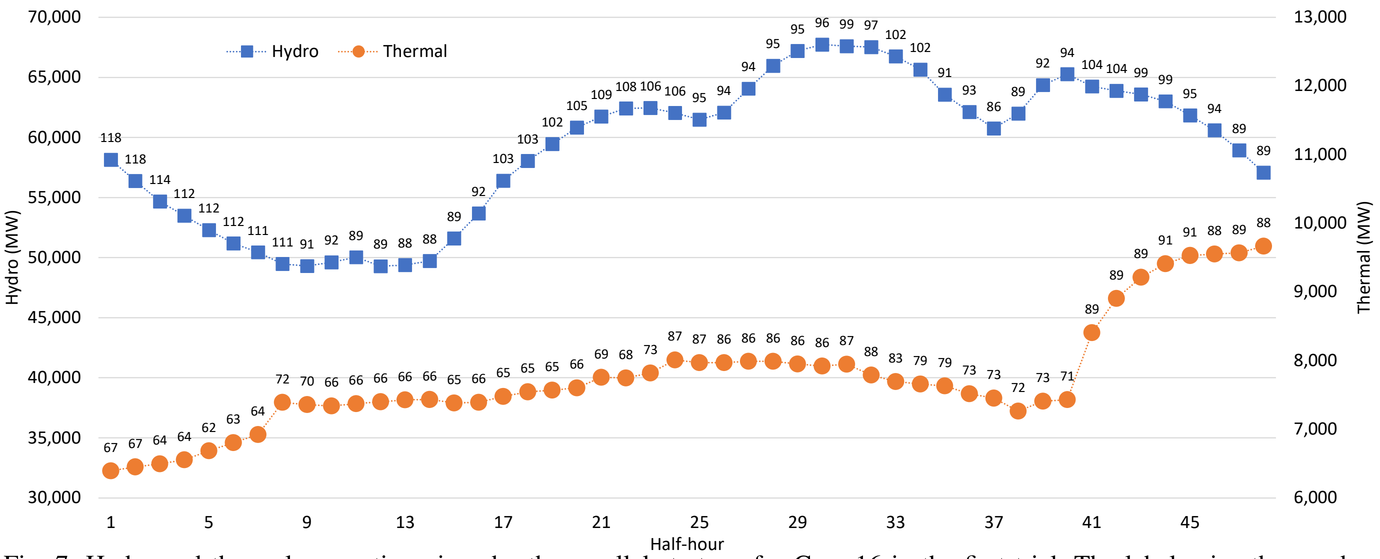


Fig. 7: Hydro and thermal generation given by the parallel strategy for Case 16 in the first trial. The labels give the number of hydro plants and thermal units committed.

[5] D. Ávila, A. Papavasiliou, and N. Löhndorf, "Parallel and distributed computing for stochastic dual dynamic programming," *Computational Management Science*, pp. 1–28, 2021.

[6] R. J. Pinto, C. T. Borges, and M. E. P. Maceira, "An efficient parallel algorithm for large scale hydrothermal system operation planning," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4888–4896, 2013.

[7] C. L. Lara, J. D. Siirola, and I. E. Grossmann, "Electric power infrastructure planning under uncertainty: stochastic dual dynamic integer programming (sddip) and parallelization scheme," *Optimization and Engineering*, vol. 21, no. 4, pp. 1243–1281, 2020.

[8] F. D. Machado, A. L. Diniz, C. L. Borges, and L. C. Brandão, "Asynchronous parallel stochastic dual dynamic programming applied to hydrothermal generation planning," *Electric Power Systems Research*, vol. 191, p. 106907, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779620307057

[9] J. Zhu, X. Mo, Y. Xia, Y. Guo, J. Chen, and M. Liu, "Fully-decentralized optimal power flow of multi-area power systems based on parallel dual dynamic programming," *IEEE Transactions on Power Systems*, pp. 1–1, 2021.

[10] F. Safdarian and A. Kargarian, "Temporal decomposition-based stochastic economic dispatch for smart grid energy management," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4544–4554, 2020.

[11] F. Safdarian, A. Mohammadi, and A. Kargarian, "Temporal decomposition for security-constrained unit commitment," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 1834–1845, 2020.

[12] F. Safdarian, A. Kargarian, and F. Hasan, "Multiclass learning-aided temporal decomposition and distributed optimization for power systems," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 4941–4952, 2021.

[13] T. N. Santos, A. L. Diniz, and C. L. T. Borges, "A new nested benders decomposition strategy for parallel processing applied to the hydrothermal scheduling problem," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1504–1512, 2017.

[14] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[15] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *Computing in Science Engineering*, vol. 23, no. 4, pp. 47–54, 2021.