A Novel Cooperative Multi-search Benders Decomposition for Solving the Hydrothermal Unit-Commitment Problem

Bruno Colonetti[1,*], Erlon Finardi[1], Paulo Larroyd[2], Felipe Beltrán[2]

Florianópolis, Brazil

Abstract

Renewable energy and modernization of power operation demand Independent System Operators (ISOs) to solve ever more complex and larger programming problems to securely and economically schedule power resources. A key step in the scheduling process is the unit commitment (UC). In a hydro-dominated system, this process also involves managing reservoirs and is called hydrothermal UC (HTUC). Due to its size, non-convex nature and strict solution-time requirements, HTUC remains a challenging problem to ISOs even in its deterministic form. Relying on a commercial solver to directly tackle this problem might require a careful configuration of the solver's parameters, and, given the generality of commercial solvers, it might not fully exploit all characteristics of the HTUC. As an alternative, researchers have recently proposed dual-decomposition strategies for addressing the UC, which enables solving subproblems in parallel. However, dualizing a non-convex problem generally results in a duality gap and requires adhoc techniques for recovering a primal solution. In this paper, we propose a novel cooperative multi-search Benders decomposition approach to solve the deterministic HTUC. Our approach benefits from being a primal method and, at the same time, it exploits the structure of the HTUC to solve subproblems in parallel. To assess the effectiveness of our proposal, we use the state-of-the-art solver Gurobi as a benchmark, and we perform our experiments on 25 cases of a large-scale system with over 1,000 generating units and 7,000 buses. On average, our approach is more than 15 times faster than Gurobi and it is able to solve the HTUC problems in under 20 min to a 0.1% gap.

Keywords: Hydrothermal unit commitment, Benders Decomposition, Cooperative multi-search
2010 MSC: 00-01, 99-00

1. Introduction

Unit commitment (UC) is a programming problem solved daily by Independent System Operators (ISO) to schedule resources in order to satisfy the day-ahead energy demand securely and economically [1]. In a centrally dispatched, cost-based system, the ISO firstly forecasts the net energy demand that is to be met by the conventional energy resources. Then, the ISO dispatches the latter resources according to their costs and operational constraints to meet the demand at the least cost. On the other hand, in a centralized, bid-based system, the ISO is responsible for scheduling the resources according to the bids from the market participants. Regardless of the market setting, UC is commonly formulated as a mixed-integer linear programming problem (MILP) in which binary variables are used for representing the status (0/1) of generating units (GU). For large systems, UC involves thousands of GUs with different characteristics scattered over vast regions. Such system configuration translates to an MILP with thousands of binary variables and hundreds of thousands (potentially millions) of constraints and continuous variables. In addition to the complexities of a large-scale MILP, the UC becomes more challenging if it includes the dispatch of hydro generation, in which case, it is called hydrothermal UC (HTUC). Hereafter, we use UC as a general term for commitment problems. While HTUC and thermal UC are similar in their purposes, dealing with hydro generation can significantly increase the computational burden: more binary variables are necessary; hydropower production functions are usually complex [2];

*Corresponding author
Email addresses: colonettib@gmail.com (Bruno Colonetti), erlon.finardi@ufsc.br (Erlon Finardi), paulo.larroyd@norus.com.br (Paulo Larroyd), felipe.beltran@norus.com.br (Felipe Beltrán)
[1]Federal University of Santa Catarina
[2]Norus

Nomenclature

Acronyms

B&B	Branch-and-bound
B&C	Branch-and-cut
BD	Benders Decomposition
CMBD	Cooperative Multi-search Benders Decomposition
GU	Generating unit
HTUC	Hydrothermal unit commitment
ISO	Independent System Operator
LB	Lower bound
MP	Master Problem
SP	Subproblem
UB	Upper bound
UC	Unit commitment

Indices and sets

$\mathscr{B}^b_g$, $\mathscr{B}^b_{h,u}$	Thermal units and hydro generating units connected to bus b, respectively
$\mathscr{G}$, $\mathscr{H}$, $\mathscr{L}$, $\mathscr{B}$	Thermal units, hydro plants, transmission lines, and network buses
$\mathscr{I}_h$	Groups of identical units of hydro plant h
$\mathscr{I}_{h,i}$	Generating units in group i of plant h
$\mathscr{M}_h$	Reservoirs upriver of hydro plant h
$\mathscr{U}_h$	Generating units of hydro plant h
g, h, u, l, b	Thermal unit, hydro plant, hydro unit, transmission line, network bus
t, i, j	time step, and auxiliary indices

Parameters

T	Number of periods in the planning horizon
n	Number of binary variables

Variables

a, b	Start up and shut down of thermal units
w	Status of hydro generating units
z, d	Status and dispatch-phase status of thermal generating units

and reservoirs introduce more temporal and spatial coupling.

Solving UC becomes even more challenging due to the short solution-times imposed: since it must be solved daily and its output must be communicated to all generation companies, the time spent on solving the problem is limited. To overcome the computational complexities and satisfactorily solve the UC in a timely fashion, researchers have recently proposed applying primal methods that mainly rely on branch-and-cut (B&C) algorithms of commercial solvers, but also include decomposition methods such as the Benders decomposition (BD), and distributed approaches based primarily on dual decomposition. Here, we refer to dual strategies as any approach that relaxes and subsequently penalizes constraint violations, e.g., Lagrangian relaxation and its many variants. In contrast, primal methods are those in which violations is not permitted. For this paper, the three most important methods under this umbrella are branch-and-bound (B&B), B&C, and BD. While a thorough overview on all methodologies proposed for solving UC is beyond the scope of this paper, we refer the interested reader to [3, 4, 5] for comprehensive reviews on this and related subjects. Here, we focus on works that propose methods for solving large-scale deterministic UC.

In [6], the authors address the UC through a decentralized augmented-Lagrangian method. In this work, the system is divided into regions, who are decoupled by the dualization of coupling constraints. Then, the dual problem is solved with the alternating-direction method of multipliers (ADMM), this enables the independent solution of the regional subproblems. Region-based decomposition and ADMM are also employed in [7], where, a release-and-fix strategy for accelerating convergence is devised. Similar ideas to [7] are employed in [8], where the authors investigate different criteria for fixing binary variables and propose strategies to reduce idle times in distributed-computing environments.

The works mentioned above share several characteristics, most important for the context of this paper is that all of them use dual decomposition. Thus, since the problem at hand is non-convex, there is no guarantee that solution found is primal-feasible. Unlike dual methods, primal methods naturally yield a primal solution (given that one can be found). Different from the previous cited works, in [9], the authors combine feasibility pump and the B&C algorithm of a commercial

solver to address the Brazilian HTUC. The capabilities of two state-of-the-art solvers are jointly exploited in [10]: the authors investigate different combinations of warm starts and solvers in an attempt to arrive at an optimal configuration. To exploit the flexibility and multitude of parameter settings of optimization solvers, [11] proposes a learning strategy for parameter configuration to speedup the solution of HTUC. BD is the method of choice in [12], where it is used to separate the problem into a scheduling problem and a network-aware one. BD is also used in [13], where the method is extended to deal with binary variables in a subproblem of the UC problem with gas-network constraints. When BD is combined with B&B, it is usually referred to as B&C (although B&C is a general term that encompasses more than this strategy). Different from the classical BD, in B&C, a single B&B tree is created and, at each node of the tree, the BD's subproblem is evaluated and an associated cut is created. In [14], such strategy is used for solving the UC with network constraints: at certain nodes of the B&B tree, the node solution is checked against network violations and, if any is found, cuts are added to the problem.

As we have seen in this brief overview, while dual methods may require primal-recovery techniques, they often break the problem into subproblems that are embarrassingly parallel, and then offer the opportunity to take advantage of the growing availability of distributed and parallel computational resources. In contrast, parallelization of deterministic problems, as opposite to, for instance, two-stage stochastic problems, with primal methods is generally dependent on the solution method. In BD, for instance, the partition of the problems yields subproblems that cannot, at first glance, be solved in parallel. Thus, in BD applied to deterministic problems, parallelization is frequently limited to the low-level algorithms solving the subproblems. This limitation is a critical disadvantage compared with the dual methods, since it may hinder the convergence of BD in a satisfactory time. Nonetheless, other primal methods offer better opportunities for parallelism. In B&B and B&C algorithms, the tasks associated with the nodes in the B&B tree can easily be solved simultaneously. In these algorithms, distinct paths of the B&B tree can be explored in parallel: the paths to be explored are created by partitioning the feasible set of the parent node according to the integrality requirement on variables. Following the terminology of [15], this can be viewed as multiple-walk parallelization, and, more specifically, as cooperative multi-search, since information from the different paths are broadcasted throughout the solution process. Note that this parallelization scheme is notably different from that obtained through a dual approach: while the parallelization in B&B and B&C algorithms are based on partitioning the entire feasible set according to the integrality of variables; dual methods explore the characteristics of the problem by relaxing coupling constraints and then yielding independent subproblems. In this paper, we devise a strategy that can exploit the characteristics of the HTUC to yield embarrassingly parallel subproblems in a BD framework. Thus, our approach aims at combining the advantages of primal and dual methods: the primal-feasibility of primal methods, and the parallelism of dual methods.

Our main contributions to the literature are: (i) a novel cooperative multi-search Benders Decomposition (CMBD); (ii) decoupling the acquisitions of upper bounds (UBs) and lower bounds (LBs) in the BD. In a deterministic setting, BD is essentially a sequential algorithm. With multi-search, we use the problem's own characteristics to enable the simultaneous exploration of several regions of its feasible set. Different from our approach, in a B&C framework, although there is a cooperative multi-search, the neighborhoods to explore are determined based on the integrality of variables with little to no regard to the characteristics of the problem. Moreover, for problems with computationally challenging subproblems, as ours, B&C might not be advisable because it can lead to the evaluation of several low-quality points, [16], which, in turn, can slow down convergence if not enough computational resources are available. In addition, (ii) is introduced to exploit a well-reported characteristic of UC: the tight bound given by its continuous relaxation. To the best of our knowledge, this characteristic has not yet been used in the way we propose. In the classical BD, an LB to the problem's optimal value is continuously, although generally rather slowly, improved after each iteration of the method. In a B&C framework, the continuously relaxation is solved prior to the beginning of the exploration of the B&B tree. Afterwards, the LB is improved as the node problems are solved, and also with the addition of valid inequalities (B&C). Thus, these two approaches are remarkably different from ours: we obtain LBs completely independently from the search for solutions. All developments in this work are specially designed and heavily based on the HTUC and its characteristics. Nonetheless, extensions of this work to other areas are possible.

The rest of the paper is organized as follows. Section 2 formally introduces the HTUC. In Sec. 3, we give an overview of BD, its main drawbacks, and the regularizing technique used in this work, as well as introduce several elements later used in the CMBD. The CMBD

is then proposed in Sec. 4. The experiments and computational settings are shown in Sec. 5, while Sec. 6 presents the results. Section 7 gives the final remarks and possible future works.

2. Problem statement

Due to the widespread knowledge of UC and for brevity, we suppress the units of variables and parameters. For better understanding, the HTUC is shown in parts.

2.1. Model of thermal generating units

We adopt the model of thermal GUs of [17]. Below, we present the constraints wherein only binary variables appear.

$$a_{g,t} - b_{g,t} = z_{g,t} - z_{g,t-1} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1a)$$

$$z_{g,t} = \sum_{i=t-\mathrm{N}^a+1}^{t} a_{g,i} + \sum_{i=t}^{t+\mathrm{N}^b-1} b_{g,i} + d_{g,t} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1b)$$

$$\sum_{i=t-\mathrm{T}^{up}+1}^{t} a_{g,i} \leq z_{g,t} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1c)$$

$$\sum_{i=t-\mathrm{T}^{down}+1}^{t} b_{g,t} \leq 1 - z_{g,t} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1d)$$

$$d_{g,t} \geq a_{g,t-\mathrm{N}^a} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1e)$$

$$a_{g,t} + b_{g,t} \leq 1 \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (1f)$$

$$\sum_{t=0}^{\mathrm{E}_g-1} z_{g,t} \geq \mathrm{E}_g \qquad \forall g \in \mathscr{G}, \quad (1g)$$

$$z_{g,t}, a_{g,t}, b_{g,t}, d_{g,t} \in \{0,1\} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}. \quad (1h)$$

In (1), logical constraints (1a) ensure that a start-up (a) or shut-down (b) reflects in a change of status (z). Similarly, (1b) dictates that, once in dispatch phase, $d_{g,t} = 1$, for arbitrary GU g and time t, the status of the GU is 1, i.e., $z_{g,t} = 1$. Otherwise, $z_{g,t}$ is only 1 if the GU is in its start-up or shut-down trajectory. Constraints (1c) and (1d) are the usual minimum up and down times, where T^{up} and T^{down} are the minimum up and down times. Upon finishing the start-up trajectory, the GU must enter its dispatch phase for a least one period of time: this is guaranteed by constraints (1e), where N^a is the number of steps in the start-up trajectory. If start up or shut down has no explicit cost, then they occur simultaneously. For this reason, constraints (1f) are added to the model. A GU whose status immediately before the beginning of the planning horizon is 1 cannot be forced to shut down for a certain amount of time, depending on its ramping capabilities. Thus, (1g) requires GU g to remain operating for at least E_g time steps. Lastly, the binary nature of a, b, d and z are enforced in (1h). (2) shows the remaining constraints and variables.

$$p_{g,t} = \sum_{i=t-\mathrm{N}^a+1}^{t} \mathrm{S}^a_{t-i} \cdot a_{g,t} + \sum_{i=t}^{t+\mathrm{N}^b-1} \mathrm{S}^d_{i-t} \cdot b_{g,2\cdot t+\mathrm{N}^b-1-i} + p^d_{g,t} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (2a)$$

$$\underline{\mathrm{P}}_g \cdot d_{g,t} \leq p^d_{g,t} \leq \overline{\mathrm{P}}_g \cdot d_{g,t} \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \quad (2b)$$

$$-\mathrm{R}^{ramp}_g \cdot d_{g,t-1} - (1 - d_{g,t-1}) \cdot \underline{\mathrm{P}}_g \leq -p^d_{g,t} + p^d_{g,t-1} \leq \mathrm{R}^{ramp}_g \cdot d_{g,t} + (1 - d_{g,t}) \cdot \underline{\mathrm{P}}_g \qquad \forall g \in \mathscr{G}, \forall t \in \mathscr{T}. \quad (2c)$$

The continuous variables associated with thermal GUs are p and p^d, respectively, the total generation and the generation in the dispatch phase. In (2), (2a) gives the total thermal generation $p_{g,t}$ for arbitrary GU g at time t: (i) $\sum_{i=t-\mathrm{N}^a+1}^{t} \mathrm{S}^a_{t-i} \cdot a_{g,t}$ accounts for the power generated at each step of the start-up trajectory; on the other hand, (ii) $\sum_{i=t}^{t+\mathrm{N}^b-1} \mathrm{S}^d_{i-t} \cdot b_{g,2\cdot t+\mathrm{N}^b-1-i}$ accounts for the power generated during the shut-down trajectory; and (iii) $p^d_{g,t}$ is the generation during dispatch phase. The start-up trajectory is defined by the number of steps, N^a, and the generation at each step, S^a. Similar parameters define the shut-down trajectories: N^b and S^b. During dispatch, the $p^d_{g,t}$, and consequently $p_{g,t}$, is bounded below and above by $\underline{\mathrm{P}}$ and $\overline{\mathrm{P}}$, as in (2b). Moreover, changes in generation during dispatch are limited by the ramp limits (2c): in this case, the increase and decrease limits are both defined by R^{ramp}. Constraints (2c) further require that the generation immediately before entering/leaving the dispatch phase must be at most the GU's minimum.

2.2. Model of the hydro generating units

In this work, we use an individual model for all hydro GUs. As opposite to the more common aggregated representation; this enables a more accurate modelling of hydro operation.

In (3), (3a) are symmetry-breaking constraints: these constraints prevent similar hydro GUs from being brought on unless an identical GU with higher priority is also on. Take, for instance, a generic hydro plant $h = 0$ with 6 GUs, i.e., $\mathscr{U}_0 = \{0, \ldots, 5\}$. Say this plant has two groups of GUs, $\mathscr{I}_0 = \{0,1\}$, such that, $\mathscr{I}_{0,0} = \{0,1,2\}$

$$w_{h,u,t} \leq w_{h,u-1,t} \qquad \forall h \in \mathscr{H}, \forall i \in \mathscr{I}_h, \forall u \in \mathscr{I}_{h,i}, \forall t \in \mathscr{T}, \quad (3a)$$

$$w_{h,u,t} \cdot \underline{\mathrm{HG}}_{h,u} \leq hg_{h,u,t} \leq w_{h,u,t} \cdot \overline{\mathrm{HG}}_{h,u} \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}, \quad (3b)$$

$$w_{h,u,t} \cdot \underline{\mathrm{Q}}_{h,u} \leq q_{h,u,t} \leq w_{h,u,t} \cdot \overline{\mathrm{Q}}_{h,u} \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}, \quad (3c)$$

$$\underline{\mathrm{V}}_h \leq v_{h,t} \leq \overline{\mathrm{V}}_h, \qquad 0 \leq s_{h,t} \leq \bar{\mathrm{S}}_h \qquad \forall h \in \mathscr{H}, \forall t \in \mathscr{T}, \quad (3d)$$

$$0 \leq hg^{pp}_{h,t} \leq \overline{\mathrm{O}}^{v}_{h,u,i} \cdot v_{h,t} + \overline{\mathrm{O}}^{q}_{h,u,i} \cdot q_{h,u,t} + \overline{\mathrm{O}}_i \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall i \in \mathscr{O}^{p,over}_{h,u}, \forall t \in \mathscr{T}, \quad (3e)$$

$$hg^{pp}_{h,t} \geq \underline{\mathrm{O}}^{v}_{h,u,i} \cdot v_{h,t} + \underline{\mathrm{O}}^{q}_{h,u,i} \cdot q_{h,u,t} + \underline{\mathrm{O}}_i \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall i \in \mathscr{O}^{p,under}_{h,u}, \forall t \in \mathscr{T}, \quad (3f)$$

$$0 \leq hg^{pl}_{h,u,t} \leq \overline{\mathrm{L}}^{q}_{h,u,i} \cdot q_{h,u,t} + \overline{\mathrm{L}}^{o}_{h,u,i} \cdot \Big(\sum_{j \in \mathscr{U} \setminus \{u\}} q_{h,j,t} + s_{h,t} \Big) + \overline{\mathrm{L}}_i \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall i \in \mathscr{O}^{l,over}_{h,u}, \forall t \in \mathscr{T}, \quad (3g)$$

$$hg^{pl}_{h,u,t} \geq \underline{\mathrm{L}}^{q}_{h,u,i} \cdot q_{h,u,t} + \underline{\mathrm{L}}^{o}_{h,u,i} \cdot \Big(\sum_{j \in \mathscr{U} \setminus \{u\}} q_{h,j,t} + s_{h,t} \Big) + \underline{\mathrm{L}}_i \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall i \in \mathscr{O}^{l,under}_{h,u}, \forall t \in \mathscr{T}, \quad (3h)$$

$$hg_{h,u,t} = hg^{pp}_{h,u,t} - hg^{pl}_{h,u,t} \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}, \quad (3i)$$

$$v_{h,t} - v_{h,t-1} + \mathrm{C}^h \cdot \Big(\sum_{u \in \mathscr{U}_h} q_{h,u,t} + s_{h,t} \Big) - \mathrm{C}^h \cdot \sum_{j \in \mathscr{M}_h} \Big(\sum_{u \in \mathscr{U}_j} q_{j,u,t-d_{j,h}} + s_{j,t-d_{j,h}} \Big) = \mathrm{C}^h \cdot \mathrm{I}_{h,t} \qquad \forall h \in \mathscr{H}, \forall i \in \mathscr{U}_h, \forall t \in \mathscr{T}, \quad (3j)$$

$$\alpha \geq \sum_{h \in \mathscr{H}} \mathrm{K}_{h,i} \cdot v_{h,\mathrm{T}-1} + \mathrm{R}^{\alpha}_i \qquad \forall i \in \mathscr{C}, \quad (3k)$$

$$w_{h,u,t} \in \{0,1\} \qquad \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}. \quad (3l)$$

and $\mathscr{I}_{0,1} = \{3,4,5\}$. Using the indices as priority measures, for the first group of GUs (GUs in $\mathscr{I}_{0,0}$), GU 1 can only be brought on if GU 0 is on. Moreover, GU 2 can only operate as long as 1 is also operating. For more on symmetry-breaking constraints, refer to [18]. Minimum and maximum generations, respectively, $\underline{\mathrm{HG}}$ and $\overline{\mathrm{HG}}$, are enforced by (3b). Inequalities (3c) constrain the turbine to be within its minimum, $\underline{\mathrm{Q}}$, and maximum, $\overline{\mathrm{Q}}$. Minimum, $\underline{\mathrm{V}}$, and maximum, $\overline{\mathrm{V}}$, reservoir volumes, v, are imposed by (3d), (3d) also limit the spillage, s, to be nonnegative and no more than its maximum $\bar{\mathrm{S}}$. In this paper, we represent the individual hydropower function for each GU as the difference between what we call potential generation, $hg^{pp}_{h,t}$, and generation loss, $hg^{pl}_{h,t}$. The former is a function of reservoir volume and turbine discharge of the GU, and it estimates the generation as if there were no head losses and the tailrace level were null. The losses in generation are accounted for by the generation-loss function through the turbine discharge of the GU, the plant's spillage and turbine discharge of other GUs. Both potential generation and generation loss are limited from below and above by piecewise linear functions. The upper limit of the potential generation is given by (3e), where $\overline{\mathrm{O}}^{v}$ is the volume coefficient, $\overline{\mathrm{O}}^{q}$ is the GU's turbine discharge coefficient, and $\overline{\mathrm{O}}$ is a constant. In turn, (3f) sets a lower bound to the potential generation in terms of the same variables and similar parameters. In (3f), the coefficients are similar to those in (3e): $\overline{\mathrm{O}}^{v}$ is the coefficient of the reservoir volume, $\overline{\mathrm{O}}^{q}$ is the coefficient of the turbine discharge, and $\overline{\mathrm{O}}$ is a constant. The upper and lower limits of the generation loss are enforced by (3g) and (3h), in which $\overline{\mathrm{L}}^{q}$ and $\underline{\mathrm{L}}^{q}$ are the GU's turbine discharge coefficient, $\overline{\mathrm{L}}^{o}$ and $\underline{\mathrm{L}}^{o}$ are the coefficients of the plant's total outflow minus the turbine discharge of GU of interest, and, finally, $\overline{\mathrm{L}}$ and $\underline{\mathrm{L}}$ are the respective constants. Water balance at the reservoirs is required by constraints (3j). In the later, C^h is a constant that converts flow into volume, $d_{j,h}$ is the water travelling time from reservoir j to h, and I is the inflow to the reservoirs. Lastly, the cost-to-go function is given in (3k) as a piecewise linear function, where α is the variable used for estimating the future cost, K are the coefficients of the reservoir volumes, and R^{α} are the constant terms in the functions.

2.3. Network

Thermal and hydro GUs are coupled by system-wide constraints: the satisfaction of the bus loads and each time step, and the limits on the line flows. The complete

network model is shown in (4).

$$\begin{aligned}
\sum_{g\in\mathscr{B}_g^b} p_{g,t} + \sum_{(h,u)\in\mathscr{B}_{h,u}^b} hg_{h,u,t} + & \\
p_{b,t}^{s,g} - p_{b,t}^{s,l} + \sum_{j\in\mathscr{B}} \mathrm{Y}_{b,j}\cdot\theta_{j,t} = \mathrm{D}_{b,t} & \quad \forall b\in\mathscr{B},\ \forall t\in\mathscr{T}, \qquad (4a)\\
0 \le p_{b,t}^{s,g}, p_{b,t}^{s,l} & \quad \forall b\in\mathscr{B},\ \forall t\in\mathscr{T}, \qquad (4b)\\
\underline{\mathrm{F}}_l \le \frac{\theta_{\mathrm{fr}_l,t} - \theta_{\mathrm{to}_l,t}}{\mathrm{r}_l} \le \bar{\mathrm{F}}_l & \quad \forall b\in\mathscr{L},\ \forall t\in\mathscr{T}. \qquad (4c)
\end{aligned}$$

The network variables are $p^{s,g}$, $p^{s,l}$, f, and θ. Variables $p^{s,g}$ and $p^{s,l}$ are the generation-shortage slack and generation-surplus slack. The flows in the transmission lines are defined by the values of the bus voltages, θ. In (4), (4a) are the power-balance constraints: generation and net flow at each bus is required to counterbalance the bus' load D; Y is the admittance matrix. Inequalities (4b) enforce the nonnegativity of the slacks $p_{b,t}^{s,g}$ and $p_{b,t}^{s,l}$. The flows in the transmission lines, as well as their upper, $\bar{\mathrm{F}}$, and lower, $\underline{\mathrm{F}}$, limits, are included in (4c), where fr_l and to_l are the from and to-bus of line l, whereas r_l is the line impedance.

2.4. Hydrothermal unit commitment

With the preceding models, we are now able to state the complete HTUC, as follows.

$$\begin{aligned}
&\min \sum_{t\in\mathscr{T}} \left[\sum_{g\in\mathscr{G}} \mathrm{C}_g\cdot p_{g,t} + \mathrm{C}^s \cdot \sum_{b\in\mathscr{B}} p_{b,t}^{s,g} + p_{b,t}^{s,l} \right] + \alpha \qquad (5)\\
&\text{s.t. (1a)-(1h), (2a)-(2c), (3a)-(3l), (4a)-(4c).}
\end{aligned}$$

The objective function in (5) consists of thermal-generation costs, C_g, costs of generation shortage and surplus, C^s, and the estimate of the future-operation cost, α. In this paper, we assume that (5) accepts at least one solution. Moreover, the objective-function coefficients C_g and C^s are strictly positive, and the cost-to-go function (3k) renders a likewise strictly positive α. Consequently, the optimal value of (5) is bounded below. In summary, (5) is a MILP with a non-empty feasible set and a bounded optimal value.

3. Benders decomposition

In the classical BD [19], the problem is partitioned based on what are defined as complicating variables. In the UC, the complicating variables are usually the binary variables — as it is the case in (5). With the partition, two problems arise: the master problem (MP), and the subproblem (SP). The MP deals with the binary variables and all constraints with only binary variables. Additionally, a cutting-plane model is used for underestimating the optimal value of the SP in the MP, and the SP's feasible region is approximated through another cutting-plane model. The SP, on the other hand, is simply the original problem with the complicating variables fixed. The classical BD algorithm works by generating iterates from the MP and subsequently evaluating them in the SP, whose information is then used for improving the cutting-plane models in the MP, and the process is repeated. The MP can be shown to be a relaxation of the original problem. Thus, its optimal value sets an LB to the original problem. Moreover, a UB can be found by evaluating an MP iterate in the SP. Once the relative optimality gap (gap), defined as $gap = (\mathrm{UB}-\mathrm{LB})/\mathrm{UB}$, reaches a value less than or equal to a pre-defined tolerance ε, the iterative process ends.

Despite its successful application in many fields, the classical BD suffers from several shortcomings [20]: iterates may wildly oscillate in successive iterations; improving the incumbent solution near optimality becomes harder due to tailing-off effects; the MP can rapidly become too time-consuming. One interesting strategy to cope with the oscillating behavior of the MP iterates and get good quality solutions fast is the regularization of the MP. Two main regularizing techniques can be employed: trust-region regularization, and level regularization. In BD, the trust-region method is closely related to the local branching of [21]. Level regularization has been mainly applied to convex continuous problems [22], but has recently been extended to convex mixed integer non-linear problems [23]. While the trust-region method searches for candidate solutions in a ball with radius R centered at a reference solution $\hat{\mathrm{x}}$, while the second searches in a level set for the closest candidate to the reference solution. In this paper, we use the trust-region regularization. Applying this regularization to the BD requires changes in the classical algorithm: one has to account for updates of R and $\hat{\mathrm{x}}$; possible infeasibility in the MP due to a over restrictive R; and, most importantly, the update of the LB. In (6),

we show the trust-region-regularized MP.

$$\begin{aligned}
\min\ & \omega && \text{(6a)}\\
\text{s.t. } & \text{(1a)-(1h)}, \text{(3a)}, \text{(3l)}, && \text{(6b)}\\
& \omega \geq f(\mathrm{x}) + \pi_{\mathrm{x}}^{f} \cdot (x - \mathrm{x}) \qquad \forall \mathrm{x} \in \mathscr{O}, && \text{(6c)}\\
& 0 \geq h(\mathrm{x}) + \pi_{x}^{h} \cdot (x - \mathrm{x}) \qquad \forall \mathrm{x} \in \mathscr{F}, && \text{(6d)}\\
& \sum_{i \in \mathscr{N}: \hat{\mathrm{x}}_i = 0} x_i + \sum_{i \in \mathscr{N}: \hat{\mathrm{x}}_i = 1} (1 - x_i) \leq \mathrm{R}, && \text{(6e)}\\
& x = (a, b, z, w), \hat{\mathrm{x}} = \left(\hat{\mathrm{a}}, \hat{\mathrm{b}}, \hat{\mathrm{z}}, \hat{\mathrm{w}}\right). && \text{(6f)}
\end{aligned}$$

The binary variables in (6) are concatenated in the n-dimensional vector x for simplicity, and its indices are stored in set $\mathscr{N} = \{0, ..., n-1\}$; similarly, the reference solution $\hat{\mathrm{x}}$ is composed of the corresponding elements of $\hat{\mathrm{a}}$, $\hat{\mathrm{b}}$, $\hat{\mathrm{z}}$, and $\hat{\mathrm{w}}$. In (6c), $f(\mathrm{x})$ is the optimal value of the SP for a previously evaluated iterate x; π_{x}^{f} is a subgradient of $f(\cdot)$ at x. Similarly, $h(\mathrm{x})$ is the optimal value of the feasibility check, introduced in the following, for iterate x and π_{x}^{h} is a subgradient of $h(\cdot)$ at x. The constraints defined by the set $\mathscr{O}$ are better known collectively as optimality cuts and they define a cutting-plane model of function $f(\cdot)$. Since $f(\cdot)$ is convex on x and π_{x}^{f} are subgradients, the cutting-plane model approximates $f(\cdot)$ from below. On the other hand, $\mathscr{F}$ forms the feasibility cuts: a collection of inequalities that forms an approximation of the domain of $f(\cdot)$. Sets $\mathscr{O}$ and $\mathscr{F}$ are populated during the iterative process: if an iterate is found to be in the domain of $f(\cdot)$, it is is added to $\mathscr{O}$; otherwise, if the particular iterate induces infeasibility to the SP, then it is added to $\mathscr{F}$. For now, it suffices to know that the feasibility of a given iterate is evaluated through the convex, linear function $h(\cdot)$.

Furthermore, in (6), $\hat{\mathrm{x}}$ is the reference solution for the trust region (6e). In this work, $\hat{\mathrm{x}}$ is required to always be a solution of (5). This implies that the trust region never induces infeasibility to the MP, since one could simply choose $\hat{\mathrm{x}}$ if no other solution inside the trust region can be found. The availability of at least one MP solution, in turn, guarantees that the trust-region-regularized MP is always feasible. Because we are dealing with binary variables and the reference solution is also 0/1, then the Euclidean distance of x and $\hat{\mathrm{x}}$ becomes simply the Hamming distance between these two vectors, which is a linear function, as in (6e). The radius of the trust region is R in (6e) and it controls how far a candidate solution may be from the current reference $\hat{\mathrm{x}}$. If R is set to a large enough value, then (6) becomes the classical, non-regularized MP. In this case, as previously mention, the optimal value of the MP sets an LB on the optimal value of the original problem. However, if R cuts part of the feasible set of the non-regularized MP, then we can no longer guarantee that the optimal value of the regularized MP produces a valid LB to the original problem since the regularized MP (6) is not a relaxation of (5).

With the MP defined, we show the SP below.

$$\begin{aligned}
f(\mathrm{x}) = \min\ & \sum_{g \in \mathscr{G}} \sum_{t \in \mathscr{T}} \mathrm{C}_g \cdot p_{g,t} + \\
& \mathrm{C}^{s} \cdot \sum_{b \in \mathscr{B}} \sum_{t \in \mathscr{T}} \left(p_{b,t}^{s,g} + p_{b,t}^{s,l} \right) + \alpha \qquad \text{(7)}\\
\text{s.t. } & \text{(2a)-(2c)}, \text{(3b)-(3k)}, \text{(4a)-(4c)},\\
& (a, b, z, w) = \mathrm{x}.
\end{aligned}$$

As previously stated, (7) is simply (5) with the binary variables fixed. For convenience and simplicity, equality $(a, b, z, w) = \mathrm{x}$ is used to convey that (a, b, z, w) take their respective values in x.

Not all iterates x generated by the MP induce feasibility in the SP — some iterates might not be in the domain of $f(\cdot)$. To prevent iterates x from inducing infeasibility in the SP, we employ two techniques: (i) the valid inequalities (1g) are added to model; (ii) a feasibility check is introduced. Valid inequalities (1g) prevent infeasibilities arising from the thermal GUs, while the feasibility check shown in the following is used to check the feasibility of x with respect to the operation of the hydro plants.

$$\begin{aligned}
h(\mathrm{x}) = \min\ & \sum_{h \in \mathscr{H}} \sum_{u \in \mathscr{U}_h} \sum_{t \in \mathscr{T}} \left(\underline{\mathrm{s}}_{h,u,t}^{q} + \overline{s}_{h,u,t}^{q} \right)\\
& + \sum_{h \in \mathscr{H}} \sum_{u \in \mathscr{U}_h} \sum_{t \in \mathscr{T}} \left(\underline{\mathrm{s}}_{h,u,t}^{hg} + \overline{s}_{h,u,t}^{hg} \right)\\
& \qquad \forall h \in \mathscr{H}, \qquad \text{(8)}\\
\text{s.t. } & 0 \leq \underline{\mathrm{s}}_{h,u,t}^{q}, \overline{s}_{h,u,t}^{q}, \underline{\mathrm{s}}_{h,u,t}^{hg}, \overline{s}_{h,u,t}^{hg} \forall u \in \mathscr{U}_h,\\
& \qquad \forall t \in \mathscr{T},\\
& \text{(3b)-(3j)}, (a, b, z, w) = \mathrm{x}.
\end{aligned}$$

In (8), the slacks $\underline{\mathrm{s}}_{h,u,t}^{q}$, $\overline{s}_{h,u,t}^{q}$, $\underline{\mathrm{s}}_{h,u,t}^{hg}$, and $\overline{s}_{h,u,t}^{hg}$ are added to constraints which are at risk of being violated. Variables $\underline{\mathrm{s}}_{h,u,t}^{q}$ and $\overline{s}_{h,u,t}^{q}$ are appropriately added to the minimum- and maximum-turbine-discharge inequalities (3c). Likewise, $\underline{\mathrm{s}}_{h,u,t}^{hg}$, and $\overline{s}_{h,u,t}^{hg}$ are included in constraints (3c). The linear and convex function $h(\cdot)$ returns zero as long as x does not induce any violations in the hydro operation in (7), otherwise $h(\mathrm{x})$ is strictly greater than zero. For the later case, a subgradient π_{x}^{h} of $h(\cdot)$ at x can be retrieved and, together with the function value $h(\mathrm{x})$ and the iterate x itself, a feasibility cut can be added to the MP to prevent x from being chosen again, as in (6d). An interesting characteristic of (8) is that it can be separated into basis, which allows the resolution of smaller problems.

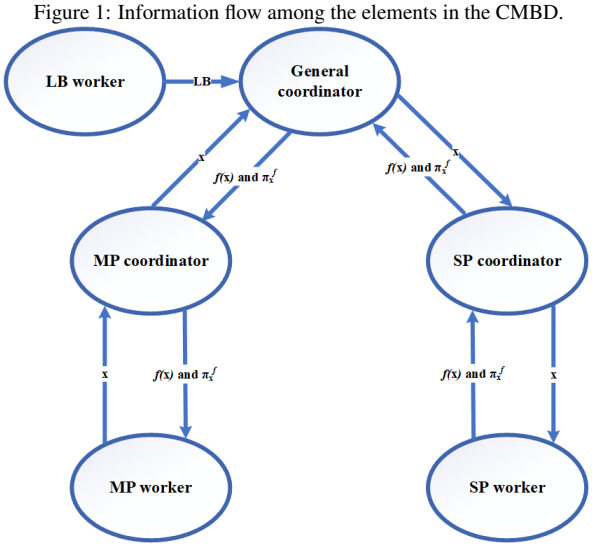
Algorithm (1) shows in a simple and compact form how the MP can be regularized with the trust-region regularization. For simplicity, possible updating rules of R are omitted. In addition, note that updating the LB at Step (4), which we intentionally do not specify, can be performed, for instance, by temporarily setting R to ∞, which, in practice, makes the trust-region constraint ineffective, and then solving the MP (6) boils down to solving the non-regularized MP.

Algorithm 1 Trust-region-regularized BD

1: $\mathscr{O} \leftarrow \emptyset$, $\mathscr{F} \leftarrow \emptyset$, $gap \leftarrow \infty$, UB $\leftarrow \infty$, LB $\leftarrow 0$, $\varepsilon > 0$, R > 0, initial reference solution $\hat{\text{x}}$
2: **while** $gap > \varepsilon$ **do**
3: Solve the regularized MP (6) and get an optimal solution x
4: Update the LB , and $gap \leftarrow (\text{UB} - \text{LB})/\text{UB}$
5: **if** $gap \leq \varepsilon$, **then exit**
6: **if** $h(\text{x}) = 0$ **then**
7: Get $f(\text{x})$ and a subgradient π^f_{x}
8: **if** $f(\text{x}) <$ UB **then**
9: $\hat{\text{x}} \leftarrow$ x, and UB $\leftarrow f(\text{x})$
10: $gap \leftarrow (\text{UB} - \text{LB})/\text{UB}$
11: **if** $gap \leq \varepsilon$, **then exit**
12: **end if**
13: $\mathscr{O} \leftarrow \mathscr{O} \cup \{\text{x}\}$
14: **else**
15: Get a subgradient π^h_{x}, and $\mathscr{F} \leftarrow \mathscr{F} \cup \{\text{x}\}$
16: **end if**
17: **end while**

The simple, sequential application of the BD in Algorithm (1) is opportune to introduce key figures that will be used in the CMBD. Evidently, all tasks described in what follows can be carried by a single entity in a sequential algorithm. Nonetheless, it is important to take advantage of the simplicity of Algorithm (1) to understand the main roles of the elements used in the CMBD. In our framework, there are three figures responsible for carrying out computations related to solving the original problem: LB worker, MP worker, and SP worker. Firstly, we have the LB worker whose sole task is to update the LB. Then, we have the MP worker which is responsible for solving an MP instance, e.g., (6); and the SP worker who evaluates the iterates given by the MP worker in the SP, i.e., the SP worker solves (7). In Algorithm (1), the LB worker performs the tasks in Step 4; the MP worker solves an MP instance in Step 3 and evaluates its viability in the SP by evaluating $h(\cdot)$ in Step 6, and then adding a feasibility cut to the MP instance if necessary; the SP worker is responsible for Step 7.

Figure 1: Information flow among the elements in the CMBD.



The coordination of these three workers and the control of information flow among them are performed by the General coordinator, the MP coordinator, and the SP coordinator. The MP coordinator receives the iterates generated by the MP worker and forwards it to the General coordinator. Afterwards, the first and second-order information retrieved at the iterate are sent by the General coordinator to the MP coordinator who then passes them to the MP worker. The role of the SP coordinator is to receive from the General coordinator the iterate to be evaluated by the SP worker. Upon receiving the iterate, the SP coordinator then sends it to the SP worker. Once the evaluation is completed, the function value and a subgradient are sent to the SP coordinator, who then transmits it to the General coordinator. The General coordinator is further responsible for updating the UB and verifying the convergence of the algorithm, i.e., checking if the gap tolerance has been satisfied. The flow of information among the workers and coordinators is summarized in Fig. 1.

Note that the framework shown in Fig. 1 enables more information other than iterates and function information to be shared among the workers and coordinators. Moreover, the roles of the MP coordinator and SP coordinator can be expanded to include a finer control of the problems solved by their respective underlings depending on the information received by these coordinators from the General coordinator.

So far, we have only vaguely discussed about updating the LB. In the next subsection, we will elaborate on our strategy for updating this bound and on the role of the LB worker previously introduced.

3.1. Lower bound

An important characteristic of Algorithm (1), which is also present in the classical, non-regularized BD, is that the updates of LB and UB are performed sequentially. More importantly, improving the LB depends on the optimality and feasibility cuts added to the MP. As a consequence, it generally takes several iterations for the BD algorithm, regularized or otherwise, to reach a satisfactory LB. In this paper, we propose a different rule for updating the LB in the BD: we exploit the tight dual bound given by the continuous relaxation of the HTUC. This bound is obtained independently from the search of a solution to the original problem. By updating the LB and UB individually, we are able to employ an appropriate algorithm for solving the continuous relaxation. As we will show empirically, the barrier algorithm solves the continuous relaxation efficiently, and it provides a tight bound. As an LB is obtained independently from the BD search, we can devise specific strategies for obtaining good quality solution with the regularized BD.

Having dealt with the LB, we now turn our attention to obtaining the first reference solution for the regularized MP. As we have assumed, the reference solution is always feasible. Nonetheless, finding a solution to an MILP can be, in itself, a arduous task. Moreover, the performance of regularized methods heavily depends on the reference solutions that are used: searching for good solutions in the neighborhood of a low-quality solution is not exactly promising. Thus, in order for our method to be successful, we not only need a feasible solution, but we need a high-quality one. To that end, we again resort to the characteristics of the HTUC.

3.2. A three-step procedure to find an initial solution

In power systems dominated by hydropower plants, the operation of the reservoirs plays an important role in the expected future operation cost. In turn, the expected future-operation cost generally far outweighs the present cost, as is the case for the Brazilian power system, for instance. As we will see in the experiments and, in fact, as it currently happens in industry, the expected future operation cost, as given by the cost-to-go function, accounts for more than 95% of the total expected cost (present and future cost). Judging by this characteristic of the problem, we can wonder whether we can devise a relaxation based on the cost-to-go function that is capable of yielding a good LB. For the HTUC shown in (5), our proposal for such relaxation is shown in (9).

$$
\begin{aligned}
\Psi = \min \quad & \alpha \\
\text{s.t. } & (3c),(3d),(3j),(3k), \\
& w_{h,u,t} \in [0,1] \ \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}.
\end{aligned} \tag{9}
$$

Optimization model (9) is nothing more than a water-only problem where we try to minimize the estimate future operation cost. Note that all constraints and variables associated with thermal GUs and the network have been neglected, as well as the hydropower production function and the integrality of variables w. Being a relaxation of (5), and given our assumption that (5) has at least one solution, and that the cost-to-go function always gives a strictly positive α, then Ψ is finite. Surprisingly, the simple, convex problem (9) is capable of providing a good LB to (5) — as we will presently show empirically. However, the importance of (9) in this paper is not to provide an LB, but rather to be the first step in the three-step procedure to find an initial solution to (5). Problem (9) is used to give an estimate of the optimal value of (5) that can then be fed to the following two steps.

Perhaps one of the first choices for trying to obtain a initial reference solution is to solve the continuous relaxation of (5), and then work towards an integral solution of (5) through some heuristics. In our experiments, however, we have found more beneficial to solve a 0-objective problem instead of the continuous relaxation: firstly, it is simpler to solve; secondly, and more importantly, it provides better initial solutions. Evidently, just setting the objective function of (5) to zero and dropping all integrality constraints would not yield good results since we would lose all sense of costs and would probably end up with a costly initial solution. Therefore, in hope of retrieving a quality solution from a 0-objective problem, we use the optimal value Ψ of (9) to bound above the objective function of (5), as shown in (10).

$$
\begin{aligned}
\min \quad & 0 \\
\text{s.t. } & (1a)\text{-}(1g),(2a)\text{-}(2c),(3a)\text{-}(3k),(4a)\text{-}(4c), \\
& \sum_{g \in \mathscr{G}} \sum_{t \in \mathscr{T}} \mathrm{C}_g \cdot p_{g,t} + \\
& \mathrm{C}^s \cdot \sum_{b \in \mathscr{B}} \sum_{t \in \mathscr{T}} \left(p^{s,g}_{b,t} + p^{s,l}_{b,t} \right) + \alpha \leq \Psi \cdot \tilde{\mathrm{N}}, \\
& z_{g,t}, a_{g,t}, b_{g,t}, d_{g,t} \in [0,1] \ \forall g \in \mathscr{G}, \forall t \in \mathscr{T}, \\
& w_{h,u,t} \in [0,1] \ \forall h \in \mathscr{H}, \forall u \in \mathscr{U}_h, \forall t \in \mathscr{T}.
\end{aligned} \tag{10}
$$

The nonemptiness of problem (5) guarantees that, as long as $\Psi \cdot \tilde{\mathrm{N}}$ is large enough, then (10) is feasible. Assuming that this is the case, once (10) is solved, we then retrieve the values of variables a, b, z, and w and round them to their respective nearest integer (either 0 or 1) to obtain $\dot{a}$, $\dot{b}$, $\dot{z}$, and $\dot{w}$. Afterwards, we project these integral vectors onto the sets of feasible integer solutions of thermal and hydro GUs. Since $\dot{a}$, $\dot{b}$, $\dot{z}$, and $\dot{w}$ are 0/1 and a, b, z, and w are binary, projecting $\dot{a}$, $\dot{b}$, $\dot{z}$, and $\dot{w}$ onto

the respective feasible sets can be written as an MILP. For thermal GUs, this projection is shown in (11).

$$\min \sum_{(g,t)\in\mathscr{G}\times\mathscr{T}:\dot{z}_{g,t}=0} z_{g,t} + \sum_{(g,t)\in\mathscr{G}\times\mathscr{T}:\dot{z}_{g,t}=1} (1-z_{g,t}) \quad (11)$$
$$\text{s.t. } (1a),(1b),(1c),(1d),(1e),(1f),(1g),(1h).$$

Only the statuses variables, i.e., z, are necessary in the objective function in (11) since they fully define all other binary thermal variables. Furthermore, problem (11) is separable into thermal GUs.

The projection of $\dot{w}$ onto the set of feasible w is given in (12).

$$\min \sum_{(h,u,t)\in\mathscr{H}\times\mathscr{U}_h\times\mathscr{T}:\dot{w}_{h,u,t}=0} w_{h,u,t} + \sum_{(h,u,t)\in\mathscr{H}\times\mathscr{U}_h\times\mathscr{T}:\dot{w}_{h,u,t}=1} \left(1-w_{h,u,t}\right) \quad (12)$$
$$\text{s.t.(3a)-(3l).}$$

Different from (11), (12) is not separable into GUs. However, we can still break (12) into basins, which considerably reduces its computational burden.

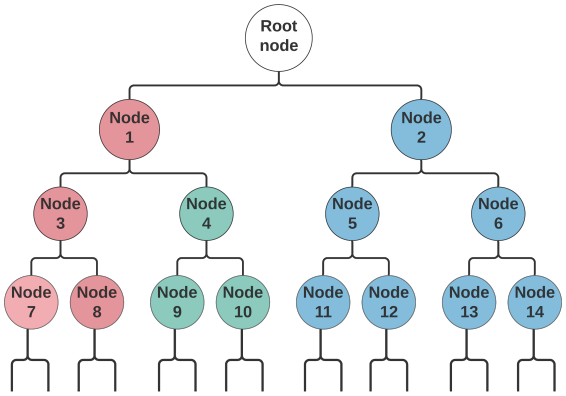
After successfully carrying out the three steps, namely, solving (9), then (10), and finally projecting the rounded variables onto the appropriate sets, we end up with a solution of (5) which will be used as the initial reference solution in the CMBD introduced next.

4. Cooperative Multi-search Benders Decomposition

According to [15], there are two classes of multi-searching: independent multi-search, and cooperative multi-search. In an independent multi-search, no information is shared during the search process. In contrast, in a cooperative multi-search, information is shared at least once before the end of the solution process.

To give a better understanding of multi-search and also to better illustrate the differences between our proposal of CMBD and the methodologies currently found in the literature, we will now use a generic B&B tree to illustrate how multi-search is used in the B&B and B&C algorithms. Figure 2 shows the cooperative multi-search exploration of a tree by three processes: red, green and blue. As we mentioned earlier, in such a tree, the subproblems are created based on the integrality of variables. Take, for instance, the root node. Once its associated problem is solved, one chooses a variable to branch on. This branching creates two subproblems, nodes 1 and 2. Thereafter, new child nodes are created through similar branching strategies. The procedure continues until a proven satisfactory solution is found by one of the processes.

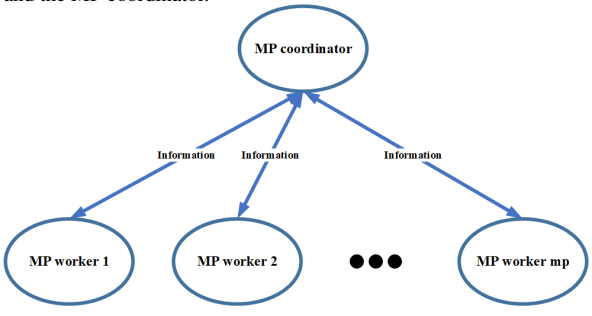
Figure 2: Example of cooperative multi-search in a B&B tree. In this case, there are three processes identified by colours red, green, and blue. For simplicity, let us imagine that one process solves the root node and shares all information with processes red and blue. Afterwards, process red solves node 1 and subsequently assigns the resolution of node 4 and all its child nodes to process green. In the meanwhile, process blue is responsible for solving the problem associated with node 2 and all child nodes created from this node.



4.1. Exploring different regions and using different algorithms with BD

Since our approach is not based on embedding BD in a B&B framework, as the works that can be found in the literature, multi-search in BD then implies generating multiple MP iterates at each iteration. In the framework previously presented, this, in turn, means having two or more MP workers solving either different MP instances and/or the same MP instances with different algorithms. From the perspective of the MP coordinator, the flow of information now becomes more complex, as more MP workers are sending and receiving information. Figure 3 illustrates this situation.

Figure 3: Illustration of the information flow between MP workers and the MP coordinator.



In order to be beneficial to the overall resolution processes, the multiple searches must take separate paths. In other words, the MP workers must use different algorithms and/or explore distinct regions. Therefore, there

are at least three alternatives for diversifying the paths: (i) exploration of different regions; (ii) same region, different algorithms; (iii) different regions, different algorithms.

In the context of BD, (i) can be achieved by exploring different regions of the MP simultaneously. For instance, consider an UC with only 4 thermal GUs, i.e., $\mathscr{G} = \{0,1,2,3\}$, an arbitrary time horizon, and two MP workers, 1 and 2. Given a reference solution $\hat{\mathrm{x}}$ and its associated components $\hat{\mathrm{a}}$, $\hat{\mathrm{b}}$ and $\hat{\mathrm{z}}$, exploring distinct regions with the two available MP workers can be achieved by setting the associated binary variables of thermal GUs 0 and 1 to their respective values in the reference solution $\hat{\mathrm{x}}$ and letting the variables associated with GUs 2 and 3 free for MP worker 1, while, for MP worker 2, variables of GUs 0 and 1 are free and those of 2 and 3 are fixed accordingly. This example is illustrated below.

MP worker 1 **MP instance 1**		**MP worker 2** **MP instance 2**	
min	ω	min	ω
s.t.	(1a) to (1h), (6c), (6f), $z_{0,t} = \hat{\mathrm{z}}_{0,t}\ \forall t \in \mathscr{T}$, $z_{1,t} = \hat{\mathrm{z}}_{1,t}\ \forall t \in \mathscr{T}$.	s.t.	(1a) to (1h), (6c), (6f), $z_{2,t} = \hat{\mathrm{z}}_{2,t}\ \forall t \in \mathscr{T}$, $z_{3,t} = \hat{\mathrm{z}}_{3,t}\ \forall t \in \mathscr{T}$.

The above example can also be used to illustrate the distinction between exploring the MP's feasible set through the strategy being discussed here and that of the B&B algorithm depicted in Fig. 2. Were we to use the B&B algorithm to partition the feasible set, one possible scenario could be the following. After solving the MP's root relaxation, with no variable fixing and regardless of the trust-region constraint, the branching that yields nodes 1 and 2 could be performed, for instance, on variable $z_{0,0}$. Then, at node 1, we would have a problem identical to the root node's except that $z_{0,0} = 0$, whereas for node 2, we would have $z_{0,0} = 1$. And, from there on, multi-search can take place. Thus, as we previously mentioned, in the B&B algorithm the paths are created based on the integrality of individual variables, while in our approach, the different paths are defined based on operational characteristics of the GUs.

In the previous example, we intentionally use MPs with no explicit regularization. However, setting some variables to their respective values in a reference solution is, in fact, an implicit regularization: by fixing the values of variables, we limit the distance between candidate solutions and the current reference solution. In addition to exploring separate regions and providing an implicit form of regularization, the strategy of fixing variables also yields more amenable MP instances because the fixed variables become redundant and can be (temporarily) removed from the problem. Note, however, that indiscriminately fixing variables may lead to regions of the MP that lack any interesting candidate solution. Lastly, fixing variables of hydro plants has to be done cautiously since the operation of hydro plants in cascades are not independent from each other. This issue will be addressed shortly.

Instead of fixing variables, if a regularizing technique such as the trust-region or the level regularization is used, exploring different regions becomes even simpler: carefully using distinct reference solutions and/or different radius (in the trust region) or level sets (in the level regularization) can lead to the exploration of different regions of the MP.

Diversifying the MP iterates can also be achieved by inspecting the same region with different algorithms (ii). An example of this approach is the resolution of the same MP, for instance, with a B&B algorithm and a metaheuristic. Since the MP generally has multiple optimal solutions, it is likely that different algorithms will not reach the same one. Most importantly, it is not always evident which algorithm is fastest, hence, tackling the same problem with distinct algorithms can also be advantageous from a solution-time point-of-view.

Finally, exploring different regions with distinct algorithms (iii) is achieved, for example, if one MP worker uses the trust-region regularization, while a second one employs the level regularization. In this case, even if the reference solution is the same, the iterates given by the regularization are likely to be different: the optimal solution of a trust-region-regularized MP tends to be on the edge of the trust region, i.e., further away from the reference solution; on the other hand, the objective function of a level-regularized MP is generally on the edge of the level set and closer to the reference solution.

In the next subsection, we elaborate on our strategies for fixing variables of hydro GUs.

4.2. Variable fixing for hydro generating units

So far we have only discussed fixing variables of thermal GUs, which is relatively straightforward since the feasibility of operation of thermal GUs is generally independent. On the other hand, for hydro GUs, fixing variables is not as simple because reservoirs are coupled through mass-balance constraints (3j). In spite of that, managing the computational burden introduced by the binary variables w is imperative to the success of the proposed method. To exemplify our approach, Fig. 4 depicts some of the most important basins in Brazil.

Figure 4: Cascade of reservoirs in Brazil. Adapted from [24].

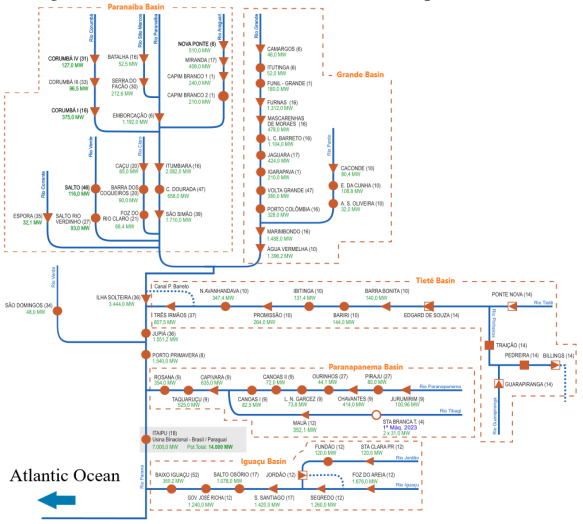


Figure 4 shows how 61 of the most significant reservoirs in the Brazilian system are connected to each other. The hydro plants shown in Fig. 4 account for over 42,000 MW of installed capacity and have about 270 hydro GUs. Given its size and complex coupling, attempting to solve an MP with all these hydro plants can be cumbersome. To overcome this difficulty, we can again exploit the characteristics of the HTUC. Note, for instance, the Iguaçu basin at the bottom of the image. Its operation does not affect the operation of any other hydro plant outside the Iguaçu basin — once water leaves the Baixo Iguaçu plant, it heads towards the Atlantic Ocean, not affecting any other hydro plant. Consequently, from a feasibility-viewpoint, the operation of the Iguaçu basin's plants and all other plants shown in Fig. 4 can be defined separately. As we can see in Fig. 4, separating the plants in the Iguaçu basin from the rest of the plants provides only a small reduction in the number of plants, and the remaining plants are all connected to each other. To break the operation of the plants shown in Fig. 4 into smaller pieces, we propose separating the basins into subbasins. From the cascades shown in Fig. 4, except from the Iguaçu Basin, we create 11 subbasins, as described in Tab. 1.

Now, take subbasin Itaipu, it is composed of a single plant: Itaipu. Because we assume a reference solution $\hat{x}$ feasible in (5) is always available, we can securely fix all other GUs in the plants in Fig. 4 to their respective values in $\hat{x}$, while keeping the variables associated with Itaipu free — similar to the example with thermal GUs. In the worst case, fixing all variables except those of Itaipu may require that the Itaipu's GUs assume their respective values in $\hat{x}$. That is, we are left with the same $\hat{w}$ that is part of $\hat{x}$. More commonly, however, the GUs of Itaipu will have enough freedom to significantly change their operation w.r.t. $\hat{x}$. For the 11 subbasins in Tab. 1 then, we can have, for example, 11 MP workers all with a single and distinct subbasin. The 11 MP workers will likely produce 11 different iterates to be later evaluated in the SP. This is a drastic difference from the classical BD, and from the B&B algorithm. Additionally, for this setting of 11 MP workers, the maximum number of hydro GUs with their statuses free to change is 72 (subbasin Grande). Consequently, the MP instances become significantly easier to solve compared to a case wherein all variables are free to change. Clearly, other arrangements than that of Tab. 1 are possible.

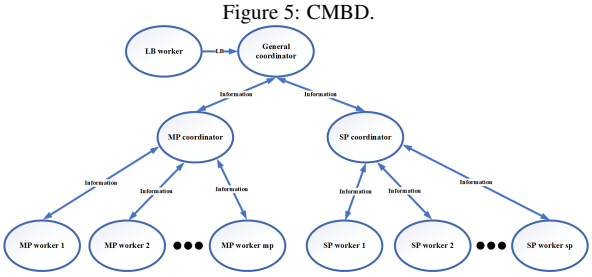
Table 1: Examples of subbasins for the cascades in Fig. 4.

Subbasin	Plants	GUs	Subbasin	Plants	GUs
Itaipu	1	20	Espora	1	3
Ilha Solteira	2	25	Verde	2	4
Paranaíba	13	49	Claro	3	6
Jupia	2	28	São Domingos	1	2
Grande	15	72	Tietê	11	30
-	-	-	Paranapanema	11	37

4.3. Cooperative multi-search in BD

At this point, we have devised a strategy for breaking the original MP into several distinct MP instances. Now we need to consider how all the MP workers are going to communicate and how the iterates generated by them are evaluated. To that end, it is helpful to first visualize in Fig. 5 the new framework of the CMBD.

Figure 5: CMBD.



The obvious difference between Fig. 5 and an equivalent figure of the classical BD is the presence of multiple MP workers. Moreover, note that, as in Fig. 3, we opt to use the general term "information" to the data being sent around between workers and coordinators. To understand the framework of the CMBD, let us start with

the tasks carried out by the MP workers. The algorithm followed by the MP workers is shown in Alg. 2.

Algorithm 2 MP worker

1: $flag \leftarrow true$, $\mathscr{R} \leftarrow \emptyset$
2: Solve (9), (10) and get $\dot{\mathrm{x}}$
3: Project $\dot{\mathrm{x}}$ on the appropriate sets to obtain $\hat{\mathrm{x}}$
4: **send** $\hat{\mathrm{x}}$ to the MP coordinator
5: **receive** $flag$, $\mathscr{R}$, $f(\mathrm{x}_i)$, $\pi^f_{x_i}$, x_i for $i \in \mathscr{R}$, and $\hat{\mathrm{x}}$ from the MP coordinator
6: $\mathscr{O} \leftarrow \mathscr{O} \cup \{\mathrm{x}_i : i \in \mathscr{R}\}$, $\mathscr{R} \leftarrow \emptyset$
7: **while** $\neg flag$ **do**
8: Solve (6) and get an optimal solution x
9: **if** $h(\mathrm{x}) = 0$ **then**
10: **send** x to the MP coordinator
11: **receive** $flag$, $\mathscr{R}$, $f(\mathrm{x}_i)$, $\pi^f_{x_i}$, x_i for $i \in \mathscr{R}$, and $\hat{\mathrm{x}}$ from the MP coordinator
12: $\mathscr{O} \leftarrow \mathscr{O} \cup \{\mathrm{x}_i : i \in \mathscr{R}\}$, $\mathscr{R} \leftarrow \emptyset$
13: **else**
14: Get a subgradient π^h_{x}, and $\mathscr{F} \leftarrow \mathscr{F} \cup \{\mathrm{x}\}$
15: **end if**
16: Update the set of free thermal GUs
17: Update the set of free hydro GUs
18: Update the trust-region radius R
19: **end while**

In Alg. 2, the first task of the MP workers is to follow the three-step procedure of Subsection 3.2 to obtain an initial reference solution $\hat{\mathrm{x}}$. In order for the MP workers to generate distinct $\hat{\mathrm{x}}$, one alternative is to use different $\tilde{\mathrm{N}}$ in (10). Once the initial reference has been obtained, it is sent to the MP coordinator. Then, the MP workers wait until information on the iterates is sent to them, Step 5. The first information received is $flag$, which is a boolean variable that indicates whether or not the process should continue. Following $flag$, the MP worker receives $\mathscr{R}$, a set containing the identifiers of the evaluated iterates and are now being received by the MP worker. After that, the worker gets the incumbent solution $\hat{\mathrm{x}}$, which thereafter serves as its reference solution. Afterwards, the MP workers start performing the more usual tasks of solving the MP and verifying the feasibility of the newly found iterate. If a feasible iterate is found, the worker sends it to its coordinator who, in turn, sends back information on evaluated iterates. Then, at Step 16, the worker updates the set of thermal GUs whose associated variables will be set equal to their corresponding values in $\hat{\mathrm{x}}$. We have stated such update in abstract manner to convey that it can be done as creatively as one wishes. For instance, one could choose to make the set of free thermal GUs change at every iteration but keep its size constant, or increase its size by carefully adding new thermal GUs based on their specific characteristics. A similar rationale is applied in Step 17, with the distinction that, for hydro GUs, we need to add or remove them by subbasins instead of individually. In Step 18, the radius can be set, for example, according to the number of free variables, or it can be set based on the iteration counter. Finally, the MP worker will continue to carry out its duties up until the moment that the boolean $flag$ sent by the MP coordinator is $false$. The flow of information from and to the MP workers is controlled by the MP coordinator, as summarized in Alg. 3

Algorithm 3 MP coordinator

1: $flag \leftarrow true$, $\mathscr{A}^{mp} \leftarrow \emptyset$, $\mathscr{S}^{mp} \leftarrow \emptyset$, $\mathscr{V}_i \forall i \in \mathscr{W}^{mp}$
2: **while** $\neg flag$ **do**
3: **while** $(|\mathscr{A}^{mp}| < \mathrm{m}^{mp})$ **do**
4: **receive** x_i from MP worker i
5: $\mathscr{S}^{mp} \leftarrow \mathscr{S}^{mp} \bigcup \{i\}$, and $\mathscr{A}^{mp} \leftarrow \mathscr{A}^{mp} \bigcup \{i\}$
6: **end while**
7: **send** x_i for $i \in \mathscr{S}^{mp}$ to the General coordinator
8: $\mathscr{S}^{mp} \leftarrow \emptyset$
9: **receive** $flag$, $\mathscr{R}^{mp}$, $f(\mathrm{x}_j)$, $\pi^f_{x_j}$, x_j and $\hat{\mathrm{x}}$ from the General coordinator
10: $\mathscr{V}_i \leftarrow \mathscr{V}_i \cup \mathscr{R}^{mp} \qquad \forall i \in \mathscr{W}^{mp}$
11: **send** $flag$, $f(\mathrm{x}_j)$, $\pi^f_{x_j}$, x_j for $j \in \mathscr{V}_i$ and $\hat{\mathrm{x}}$ to the MP workers i in $\mathscr{A}^{mp}$
12: $\mathscr{V}_i \leftarrow \emptyset \qquad \forall i \in \mathscr{A}^{mp}$, and $\mathscr{A}^{mp} \leftarrow \emptyset$
13: **end while**
14: **send** $flag$ to all MP workers

In Alg. 3, the MP coordinator keeps track of the available MP workers by populating the set $\mathscr{A}^{mp}$, if a given worker is not in this set, then its currently solving its corresponding MP instance. The set of MP workers is given in $\mathscr{W}^{mp}$. Moreover, the iterates sent by the MP workers are uniquely identified in set $\mathscr{S}^{mp}$. At Step 3 of Alg. 3, the MP coordinator will wait until at least m^{mp} iterates are sent from the MP workers. Here, there are at least two possibilities: (i) m^{mp} can be set to $|\mathscr{W}^{mp}|$, (ii) m^{mp} can be set to 1. In (i), the computations of the MP workers are synchronized: the MP coordinator has to wait until all MP workers finish their tasks in order to proceed. On the other hand, by relaxing m^{mp} to 1, we have an asynchronous algorithm. Once the minimum number of received iterates m^{mp} has been satisfied, they are sent to the General coordinator. The later returns information on the iterates identified in set $\mathscr{R}^{mp}$. The content of $\mathscr{R}^{mp}$ will depend on the existence of synchronization points. If the algorithm is set to its synchronous

version, then $\mathscr{R}^{mp}$ contains all points previously sent to the General coordinator. In contrast, if there are no synchronization points, then the information received from the General coordinator might not be on all iterates just sent to the General coordinator, it might be on 'older' iterates. In our work, cooperation is achieved by sharing optimality cuts and the incumbent solution among all MP workers. Given the possibility of asynchronicity, the MP coordinator has to keep track of what information each of the MP workers has yet to receive. Thus, sets $\mathscr{V}_i$ are used as storage of the optimality cuts that have yet to be sent to each of the workers. Naturally, in a synchronous algorithm, all sets $\mathscr{V}_i$ are all identical and, consequently, all MP workers enjoy the same information. A different strategy is used for the incumbent solution $\hat{\mathrm{x}}$: we have found it advantageous to always send the best solution found so far to all MP workers as soon as they become available.

As previously mentioned, the iterates generated by the MP workers are evaluated by SP workers. The tasks of these workers are summarized in Alg. 4.

Algorithm 4 SP worker

$flag \leftarrow true$
receive $flag$ and x from the SP coordinator
while $\neg flag$ **do**
    Evaluate $f(\cdot)$ at x and obtain $f(\mathrm{x})$ and π_x^f
    send $f(\mathrm{x})$, x and π_x^f to the SP coordinator
    receive $flag$ and x from the SP coordinator
end while

As we can see in Alg. 4, the tasks performed by the SP workers are rather simple: they take a single iterate x from the SP coordinator, obtain the function value $f(\mathrm{x})$ and one subgradient π_x^f at point x, and later send these information to their coordinator. The SP coordinator, on the other hand, controls the iterates being sent to the SP workers, as shown in Alg. 5.

The iterates to be evaluated in the SP are identified in set $\mathscr{R}^{sp}$. These iterates are then stored in the queue of iterates $\mathscr{Q}$. In turn, the iterates in this queue are sent to the available SP workers $\mathscr{A}^{sp}$. Note that the number of iterates yet not evaluated in queue $\mathscr{Q}$ might not be the same as the number of available workers $|\mathscr{A}^{sp}|$. Thus, after distributing the tasks, the set of available workers has to be updated by simply removing from it the indices of workers who have received a new task. In accordance with the MP coordinator, the SP coordinator waits until at least m^{sp} SP workers have responded. The newly available information is then stored in $\mathscr{S}^{sp}$, and later sent to the General coordinator.

Algorithm 5 SP coordinator

$flag \leftarrow true$, $\mathscr{A}^{sp} \leftarrow \emptyset$, $\mathscr{S}^{sp} \leftarrow \emptyset$, $\mathscr{Q} \leftarrow \emptyset$
receive $flag$, $\mathscr{R}^{sp}$, and x_j from the General coordinator
$\mathscr{Q} \leftarrow \mathscr{Q} \cup \{\mathrm{x}_j : j \in \mathscr{R}^{sp}\}$
send $flag$, x_j for $j \in \mathscr{Q}$ to the workers in $\mathscr{A}^{sp}$
Update $\mathscr{A}^{sp}$ accordingly
while $\neg flag$ **do**
    while $(|\mathscr{A}^{sp}| < \mathrm{m}^{sp})$ **do**
        receive $f(\mathrm{x}_i)$, x_i and $\pi_{x_i}^f$ from SP worker i
        $\mathscr{S}^{sp} \leftarrow \mathscr{S}^{sp} \bigcup \{i\}$, $\mathscr{A}^{sp} \leftarrow \mathscr{A}^{sp} \bigcup \{i\}$
    end while
    send $\mathscr{S}^{sp}$, $f(\mathrm{x}_i)$, x_i and $\pi_{x_i}^f$ for $i \in \mathscr{S}^{sp}$ to the General coordinator
    $\mathscr{S}^{sp} \leftarrow \emptyset$
    receive $flag$, $\mathscr{R}^{sp}$, and x_j from the General coordinator
    $\mathscr{Q} \leftarrow \mathscr{Q} \cup \{\mathrm{x}_j : j \in \mathscr{R}^{sp}\}$
    send $flag$, x_j for $j \in \mathscr{Q}$ to the workers in $\mathscr{A}^{sp}$
    Update $\mathscr{A}^{sp}$ accordingly
end while
send $flag$ to all SP workers

Lastly, the responsibilities of the General coordinator are highlighted in Alg. 6.

The General coordinator communicates with three figures: MP coordinator, SP coordinator, and LB worker. Although the job performed by the LB worker is essential to the convergence of the overall algorithm, it can be briefly summarized as solving the continuous relaxation of (5) and communicating the LB found to the General coordinator. As for the interactions of the General coordinator with the MP and SP coordinators, it can be simply seen as a bridge between the two former coordinators. In this context, the most important task carried out by the General coordinator is checking the convergence of the overall algorithm and the quality of the iterates evaluated in the SP.

With the CMBD presented above, we now proceed to introducing the computational setting used in this work, the experiments and their associated results in the following section.

5. Computational experiments

The test system is based on the Brazilian system, and it has 7,475 buses, 10,702 transmission lines, 161 reservoirs, 743 hydro GUs, and 329 thermal GUs. The total installed capacity of hydro and thermals GUs is 133,823 MW, from which 109,921 MW comes from the hydro

Algorithm 6 General coordinator

$flag \leftarrow true$, $\varepsilon > 0$
while $\neg flag$ **do**
  receive *msg*
  if *msg* is from MP coordinator **then**
    receive $\mathscr{S}^{mp}$ and x_i for $i \in \mathscr{S}^{mp}$ from the MP coordinator
  else if *msg* is from SP coordinator **then**
    receive $\mathscr{S}^{sp}$, $f(\mathrm{x}_i)$, x_i and $\pi^f_{x_i}$ for $i \in \mathscr{S}^{sp}$ from the SP coordinator
  else
    receive LB from LB worker
    $gap \leftarrow (\mathrm{UB} - \mathrm{LB})/\mathrm{UB}$
    if $gap \leq \varepsilon$, **then** $flag \leftarrow false$
  end if
  for $i \in \mathscr{S}^{sp}$ **do**
    if $f(\mathrm{x}_i) < \mathrm{UB}$ **then**
      $\hat{\mathrm{x}} \leftarrow \mathrm{x}_i$, $\mathrm{UB} \leftarrow f(\mathrm{x}_i)$
      $gap \leftarrow (\mathrm{UB} - \mathrm{LB})/\mathrm{UB}$
      if $gap \leq \varepsilon$, **then** $flag \leftarrow false$
    end if
  end for
  send $flag$, $\mathscr{S}^{sp}$, and x_j to the SP coordinator
  $\mathscr{S}^{sp} \leftarrow \emptyset$
  receive $flag$, $\mathscr{S}^{mp}$, $f(\mathrm{x}_j)$, $\pi^f_{x_j}$, x_j and $\hat{\mathrm{x}}$ to the MP coordinator
  $\mathscr{S}^{mp} \leftarrow \emptyset$
end while

GUs, while thermal GUs account for 23,902.2 MW. The test bed consists of 25 cases, all of them with different inflows, loads and initial conditions. We hereafter refer to them as Case 1 to Case 25. The planning horizon is set to 24 h with a half-hour discretization (48 periods). Table 2 shows the problem size of the original, non-decomposed HTUC considering all components of the system, i.e., (5).

Table 2: Size of the original, non-decomposed HTUC.

Constraints	Continuous vars.	Binary vars.
1,397,974	1,285,905	119,888

For the Brazilian system, 15 basins such as that of Fig. 4 (although no other is as large) can be identified. Following the discussion in Subsection 4.2, we separate the large 15 basins into the 42 subbasins shown in Tab. 3. Although other arrangements are possible and could potentially be more beneficial from a computational perspective, all of our experiments with BD are conducted using those 42 subbasins.

We run all experiments on a single machine with 2 2.60-GHz Intel Xeon E5-2660 processors, a total of 20 physical cores (10 in each process), and 128 GB of RAM. Optimization models are solved with Gurobi 9.1.1 [25], while inter-process communication is done through Microsoft's Message Passing Interface implementation, MS-MPI. We use Python 3.7.4 as our programming language; mpi4py [26] interfaces Python and MS-MPI, and Windows Server 2016 Datacenter is the operating system. All times present in the following results are wall-clock times.

The main algorithm proposed in Section 4 takes one of two forms: synchronous or asynchronous. Due to the computational-load unbalance between MP workers and SP workers, our experiments have shown that an asynchronous algorithm delivers better results than its synchronous counterpart. Thus, in this work, we choose to only present results from the asynchronous CMBD. In spite of that, we still refer to this asynchronous version simply as CMBD. Because of the inherently uncertain paths taken by the MP workers in an asynchronous algorithm, we run each case 10 times in order to obtain a large-enough data sample, which we call trials. Note that, the 10 trials might not be enough to give a statistically satisfactory average running time: they are only meant to provide a reasonable support to our claim that the CMBD can solve the problem at hand efficiently. In what follows, we refer to a specific trial of a given case simply as, for instance, Case 1, Trial 10.

The rules for updating the sets of free thermal GUs and hydro GUs' variables, and the rules for updating the trust-region radius R in Alg. 2 are crucial for the success of the overall CMBD. In this paper, we use the rules described in Tab. 4 for all MP workers. Moreover, these same rules are used for all 25 cases. While the strong results presented in the next section suggest that these rules work, on average, well, one could certainly modify them according to specific conditions of the system and/or computational resources.

According to the MP worker configurations in Tab. 4, the MP instances start essentially as the classical BD in which some variables are fixed to their values in the current reference solution. Based our experiments, restricting the search to a small trust region at the beginning of the solving process, when the cutting-plane models are still poor, is usually not beneficial because what tends to happen is that a too optimistic solution is chosen. By allowing only a handful of GUs to freely

Table 3: Separation of the Brazilian basins into subbasins, and their respective total installed capacity, and total number of GUs.

Subbasin	Basin	Installed cap. (MW)	GUs	Subbasin	Basin	Installed cap. (MW)	GUs
Grande	Paraná	7,546.50	72	Araguari	Araguari	549.00	9
Paranaíba	Paraná	8,078.60	49	Capivari	Capivari	274.40	4
Tocantins	Tocantins	12,991.65	49	Jari	Jari	389.70	3
Iguaçu	Iguaçu	7,268.00	29	Paraguaçu	Paraguaçu	160.00	2
Paranapanema	Paraná	2,713.20	37	Uatumã	Uatumã	250.00	5
Uruguai	Uruguai	5,625.64	28	São Domingos	Paraná	48.00	2
Jupia	Paraná	3,091.20	28	Ijuí	Ijuí	128.00	4
Ilha Solteira	Paraná	4,251.50	25	Parnaíba	Parnaíba	237.10	4
Tietê	Paraná	1,915.80	30	Manso	Manso	210.00	4
Itaipu	Paraná	14,000.00	20	Espora	Paraná	32.10	3
Xingú	Xingú	12,131.10	24	Itajaí	Itajaí	191.88	2
Piracicaba	Piracicaba	1,229.60	28	Jamari	Jamari	216.00	5
Madeira	Madeira	7,296.00	100	Guaporé	Guaporé	120.00	3
São Francisco	São Francisco	10,580.70	52	Jauru	Jauru	121.50	3
Paraíba do Sul	Paraíba do Sul	1,616.54	33	Itiquira	Itiquira	157.40	4
Jacuí	Jacuí	963.00	14	Mucuri	Mucuri	61.80	3
Teles Pires	Teles Pires	3,222.00	14	Aripuanã	Aripuanã	73.20	3
Claro	Paraná	223.40	6	Itabapoana	Itabapoana	55.00	2
Jequitinhonha	Jequitinhonha	861.00	6	Correntes	Correntes	176.10	3
Verde	Paraná	209.00	4	Coruá-Una	Coruá-Una	30.30	3
Taquari-Antas	Taquari-Antas	364.29	7	Comemoração	Comemoração	261.00	5

Table 4: Rules for updating the trust-region radius, set of free hydro GUs, and set of free thermal GUs for all MP workers.

Parameter	Initial condition	Update	Max.	Min.
R	20,000	Decrease 20% every iteration	20,000	50
Set of free hydro GUs	0	Add 1 random subbasin	10 subbasins	0
Set of free thermal GUs	30 randomly chosen GUs	Add 30 random GUs	∞	0

change, we can establish a trade-off between the optimism of the still poor cutting-plane model of $f(\cdot)$ and the conservativeness of a regularization. As the algorithm progresses and the cutting-plane models are improved, more GUs, both thermal and hydro, are added to the set of free GUs and the radius of the trust region is slowly decreased. Thus, we iteratively walk from the classical BD to the trust-region-regularized BD, both of them with fixed variables.

Table 4 shows that, while we add 1 new subbasin every iteration, the total number of subbasins is limited to 10. This is necessary to prevent the MPs from becoming too complex to be solved in a timely fashion. On the other hand, we have found advantageous to let all thermal GUs be added to the set of free GUs because they introduce relatively less complexity than the hydro GUs. Lastly, we have the randomness mentioned in Tab. 4. First, let us assume that we have a suboptimal solution of the HTUC, say, x. Now, what changes in x should we make in order to arrive at a better solution? In the same sense, let us assume that we currently have reference solution $\hat{x}$, and, we have a set of 30 thermal GUs that are free to charge. What GUs should we add to the set of free GUs in order to get the best possible improvement at the subsequent iteration? Evidently, owing to the non-convexity and non-differentiability of the HTUC, at this time, it is impossible to know for certain what changes we should make. In our investigations, we have concluded that the best approach, on average, is to choose the set of free GUs randomly. Although these choices are random, we set different seeds for each of the MP workers, so there is a high probability that the sets chosen by them will most likely not be the same. Moreover, as Tab. 4 indicates, at every iteration, the set of free GUs is completely randomized which likely leads significantly different sets. For instance, suppose that a given MP worker now has 5 subbasins and 150 thermal GUs, then, in the subsequent iteration, this worker

will randomly choose 6 subbasins from the 42 of Tab. 3 and 60 thermal GUs. The newly chosen subbasins and thermal GUs might be completely different from those of the previous iteration. Finally, note that, albeit we choose the sets randomly, the number of MP workers increases our chances of selecting promising sets.

For the CMBD, we use 8 MP workers and 8 SP workers. Each MP worker and the LB worker are only allowed to take a single thread for solving their respective problems, while the SP workers can use up to 4 threads for solving the SP. The gap tolerance is set to 0.1%, while the time limit for this method is set to 30 min. Given the limited computational resources, we use only one MP worker for obtaining an initial reference solution. This worker is allowed to use all threads while solving (10). In all experiments, we set the constant $\tilde{N}$ in (10) to 1.03; in other words, the original costs are bounded above by the optimal value of problem (9), Ψ, plus 3% of Ψ.

When solving the HTUC directly with Gurobi, we leave Gurobi at its default settings, with the exception of the root-relaxation method, which we set to the barrier method. The gap tolerance is set to 0.1%, while the time limit is 3 h. The 0.1% gap tolerance is utilized by at least two major system operators: the Brazilian system operator [9]; and the Midcontinent Independent System Operator [10]. We use Gurobi for two tests: in the first, Gurobi is given no initial solution; in the second, Gurobi is given the same initial solution obtained through the three-step procedure of Subsection 3.2.

The results for the test cases and strategies presented in this section are given in the following.

6. Results

The first results we show are related to the LB on the optimal value of the HTUC, and also the results given by Gurobi without an initial solution. Table 5 shows the dual bounds obtained by solving the water-only problem (9), the LBs at different stages of Gurobi's B&C algorithm, and the LBs sent by the LB worker and their associated times. Since Gurobi is not able to find any solution of the HTUC in this setting, we omit UBs and gaps from Tab. 5. The fact that Gurobi is not able to find a solution should not be interpreted as an inability of Gurobi in solving the problem at hand. It can, however, be used to attest the difficulty of the HTUC. Table 5 shows interesting results related to the aforementioned characteristics of the HTUC: specifically, the dominance of the future cost, and the tight bound given by the continuous relaxation. Note that the optimal value of problem (9) gives an LB that is over 99.00% of the LB given by Gurobi after 3 h. This is clearly due to the dominance of the future operation cost in the total cost of the HTUC. Additionally, we can see in Tab. 5 that the continuous relaxation of the HTUC yields an LB that is more than 99.99% of the final LB obtained by Gurobi after 3 h in all cases. As we will see shortly, the LBs shown in Tab. 5 are indeed very close to the optimal value, and they are sufficiently good for the pre-defined gap tolerance of 0.1%. These results are the main motivation for our proposal of Subsection 3.1 to decouple the acquisitions of UBs and LBs.

Still on the LB but now in the context of CMBD, Tab. 5 also presents the LBs delivered by the LB worker for one trial of each case under consideration. According to these results, the continuous relaxation is solved to optimality well before the time limit for all cases. Then, a tight LB is readily available to the General coordinator. As we will see in the following results, the LB worker continuously sends updates of the LB to the General coordinator. For instance, if during the barrier phase the dual and primal residuals are small enough, then the LB worker sends the dual objective value to the General coordinator, as it is a valid LB to the HTUC. By doing so, the convergence of the overall algorithm becomes less dependent on the LB worker, and we better exploit the characteristics of the problem formulation.

After discussing the LB given by the continuous relaxation, we show the main results of the three-step procedure presented in Subsection 3.2. In Tab. 6, we present the average times of each step over all cases. We care to note that these times do not significantly vary from case to case.

Table 6: Running times of the three-step procedure.

Problem	**Time (sec)**
(9)	5
(10)	90
(11), (12)	60
Total	155

According to Tab. 6, nearly 60% of the total time is due to problem (10). Despite the presence of binary variables, the separability of the projections (11) and (12) yields problems that are more amenable and can be solved efficiently. Lastly, the small, convex, linear problem (9) is easily solved by Gurobi. Table 7 then presents the UB given by the three-step procedure for each case. Associated with the UB of each case, we show the gap relative to the best LB found by Gurobi (column B&C) in Tab. 5.

Table 5: Column 2 shows the LBs given by (9); columns 3-7 present the LBs in different stages of Gurobi's B&C algorithm. More specifically, Column RL - Barrier presents the LB produced by the barrier algorithm applied to the root relaxation (RL) of (5). Column RL - Crossover shows the LB after a basic solution is found from the mid-face solution given by barrier. The fifth column presents the final LB given by Gurobi's B&C algorithm (B&C) after 3 h. Finally, columns 6 and 7 present the % ratio of the LB given by (9) and the final LB of B&C, and the LB given by the barrier algorithm applied to the RL and the final LB of B&C, respectively. Different from columns 1-7, columns 8-9 concern results from CMBD. Column 8 gives the LB delivered by the LB worker, while column 9 shows the total time, i.e., time spent in barrier plus the crossover time, taken by the LB worker to solve the continuous relaxation.

		Gurobi with no initial solution					**LB worker**	
Case	(9) (10^6\$)	**LB-RL Barrier** (10^6\$)	**LB-RL Crossover** (10^6\$)	**LB B&C** (10^6\$)	**(Col. 2) /(Col. 5)**	**(Col. 3) /(Col. 5)**	**LB** (10^6\$)	**Time (sec)**
1	59,836.42	60,307.74	60,307.74	60,308.40	99.217%	99.999%	60,307.73	389.3
2	60,037.41	60,436.77	60,436.77	60,437.43	99.338%	99.999%	60,436.75	544.07
3	59,910.57	60,230.86	60,230.86	60,231.76	99.467%	99.999%	60,230.85	360.24
4	60,062.09	60,289.63	60,289.63	60,290.38	99.621%	99.999%	60,289.62	410.39
5	60,128.58	60,269.51	60,269.51	60,270.13	99.765%	99.999%	60,269.48	375.53
6	56,445.42	56,839.95	56,839.95	56,840.65	99.305%	99.999%	56,839.90	322.42
7	56,605.17	56,934.19	56,934.19	56,935.06	99.421%	99.998%	56,934.18	628.36
8	56,789.23	57,043.38	57,043.38	57,044.32	99.553%	99.998%	57,043.36	389.55
9	56,984.24	57,169.51	57,169.51	57,170.37	99.674%	99.998%	57,169.49	309.25
10	57,154.94	57,262.08	57,262.08	57,263.04	99.811%	99.998%	57,262.04	311.28
11	119,771.30	120,385.50	120,385.50	120,386.50	99.489%	99.999%	120,385.53	490.88
12	119,755.80	120,257.90	120,257.90	120,258.70	99.582%	99.999%	120,257.93	418.13
13	119,947.20	120,336.60	120,336.60	120,337.30	99.676%	99.999%	120,336.54	606.08
14	120,112.70	120,388.50	120,388.50	120,389.40	99.770%	99.999%	120,388.54	738.88
15	120,211.90	120,375.50	120,375.50	120,376.10	99.864%	99.999%	120,375.50	352.85
16	82,418.44	82,904.66	82,904.66	82,905.69	99.412%	99.999%	82,904.63	380.42
17	82,479.99	82,874.42	82,874.42	82,875.53	99.523%	99.999%	82,874.42	355.47
18	82,607.92	82,927.66	82,927.66	82,928.44	99.614%	99.999%	82,927.65	363.13
19	82,922.92	83,133.47	83,133.47	83,134.23	99.746%	99.999%	83,133.46	396.03
20	83,052.09	83,176.77	83,176.77	83,177.53	99.849%	99.999%	83,176.75	298.33
21	59,654.60	60,022.58	60,022.58	60,023.52	99.385%	99.998%	60,022.57	381.1
22	59,749.02	60,064.85	60,064.85	60,065.62	99.473%	99.999%	60,064.83	396.07
23	59,890.50	60,140.60	60,140.60	60,141.40	99.583%	99.999%	60,140.58	436.3
24	60,015.19	60,201.79	60,201.79	60,202.67	99.689%	99.999%	60,201.78	588.46
25	60,141.31	60,252.21	60,252.21	60,253.10	99.814%	99.999%	60,252.21	328.38

Table 7: UB given by the initial solution obtained through the three-step procedure of Subsection 3.2 and the corresponding gap relative to the best LB found by Gurobi after 3 h in Tab. 5.

Case	**UB** (10^6\$)	**Gap** (%)	**Case**	**UB** (10^6\$)	**Gap** (%)
1	60,573.36	0.44	13	122,682.21	1.91
2	60,730.46	0.48	14	121,969.68	1.30
3	60,473.32	0.40	15	122,207.49	1.50
4	60,619.85	0.54	16	83,350.65	0.53
5	60,691.71	0.70	17	83,419.51	0.65
6	57,076.53	0.41	18	83,404.96	0.57
7	57,412.32	0.83	19	83,558.69	0.51
8	57,284.93	0.42	20	83,569.91	0.47
9	57,368.64	0.35	21	60,354.73	0.55
10	57,417.10	0.27	22	60,331.78	0.44
11	122,061.58	1.37	23	60,383.58	0.40
12	121,231.11	0.80	24	60,420.42	0.36
-	-	-	25	60,428.92	0.29

Table 7 shows that the gap of the initial solutions range from 0.268% to 1.911%, with an average of 0.66%, and, in most cases, it is well below 1%. The apparent difficulty of Gurobi in finding an integral solution of the HTUC, and the relatively small gaps given by the initial solutions corroborate the quality of the method proposed in Subsection 3.2.

As we have previously discussed, without an initial solution, Gurobi is not able to find any solution of the HTUC after 3 h. A natural next step it to supply Gurobi with the solutions delivered by the method of Subsection 3.2. As a consequence of that, the performance of Gurobi is significantly improved, as expected. The first 5 columns of Tab. 8 shows the results of this approach for each case.

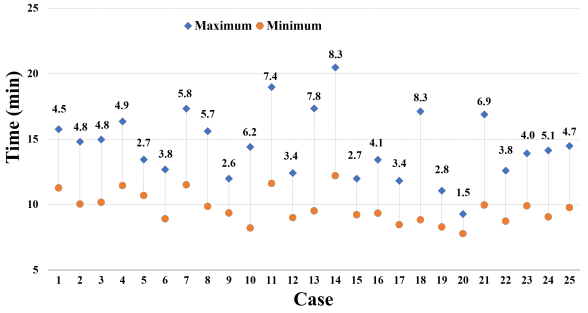
Equipped with an initial solution, Gurobi is now able to reach the desired gap tolerance of 0.1% for 4 cases: 3, 12, 14 and 20. Nonetheless, the least running time (Case 3) is just about 1 h and 40 min, while the longest running time for a convergent case is nearly 3 h, Case 12. For all other cases, by comparing Tables 7 and 8, we can observe that Gurobi is not able to significantly improve the initial solution. Again, these results are further evidence of the difficulty of the problem.

Having attested the difficulty of the problem at hand, and having empirically shown the tight bound provided by the continuous relaxation and the quality of the initial solution delivered by the proposed three-step procedure, we now present the results of the CMBD. Columns 9-11 of Tab. 8 detail the main results for the CMBD.

Examining Table 8 readily reveals the computational benefits of CMBD: in contrast with Gurobi, CMBD is able to achieve the desired gap, on average, well before the 30-min time limit. CMBD, on average, converges to a 0.1% solution more than 15 times faster than Gurobi. A case-by-case comparison shows that, on average, the improvements vary from 8.15 faster in favor of CMBD, for Case 3, to over 18 faster, again in favor of CMBD, for Case 20. Perhaps more importantly, the results reported indicate that CMBD can consistently reach the desired gap tolerance before 20 min: the arithmetic average over all cases and trials is 12 min, which is a remarkable time for a problem of the size and complexity of the HTUC. It is also evident from Tab. 8 that our strategy of decoupling the UB and the LB is very suitable for the HTUC, since the LB obtained by the LB worker is tight enough for all cases. Additionally, note that CMBD is efficient in reducing the gap to near optimality: one of the most reported drawbacks of the classical BD is the tailing-off effect [20], i.e., the difficulty of a cutting-plane-based method in reducing the gap when close to optimality. Lastly, it is also interesting to note the improvements in the gap provided by CMBD over Gurobi. On average, CMBD delivers a final gap that is more than 5 times smaller than Gurobi's. For Case 15, while Gurobi ends with a gap of 1.489%, CMBD's gap is 0.089%. On the other hand, for Cases 3, 12, 14 and 20, Gurobi delivers better gaps.

Despite the strong results reported in Tab. 8, the inherent asynchronicity of CMBD can lead to differences between running times of the same problem. To analyze these differences for the problem at hand, we present in Fig. 6 the maximums and minimums of each case over all 10 trials.

Figure 6: Maximum and minimum running times in minutes over the 10 trials of each case. The difference of the maximum running time and the minimum are given in boldface above the respective maximum.



In Fig. 6, we can see that while some cases suffer variations from minimum to maximum of more than 8 min, for instance, Case 14, other cases almost show no difference, for instance, Case 20 presents a difference of only 1.5 min. It is interesting to observe, by comparing Tab. 7 with Fig. 6, that there is no apparent correlation between a relatively poor initial solution and the differences between maximum and minimum times reported in Fig. 6. Moreover, Fig. 6 shows that the maximum running time over all cases and trials occurs for Case 14, for which one of the trials takes about 20.5 min. While it is evident from Fig. 6 that 10 trials are far from enough to provide statistically significant results, they do strongly indicate the good performance of CMBD: all cases are solved well before the 30-min time limit can be reached. Furthermore, in our investigations, we have found that these differences in running time are mainly due to distinct paths taken by the algorithms, which are caused, for instance, by unmanageable differences in the running times of MP and SP instances. To give a better understanding of the source of the running times differences presented in Fig. 6, let us take Cases 14 and 20 as examples. Table 9 shows the running time and the number of improving solutions for Cases 14 and 20. Improving solutions are those strictly better than the incumbent — in Alg. 6, an improving solution satisfies the condition of Step 14.

Table 8: Columns 2-5 present results associated with Gurobi, while Columns 6-9 show the averages of the UBs, LBs, gaps and times over all 10 trials for the CMBD in each case. The average of a particular metric is given simply as the arithmetic average over the 10 trials. Columns 10-11 offer a comparison of CMBD and Gurobi: Column 10 is the ratio of Gurobi's Time, Column 5, and the average CMBD's time, Column 9; Column 11 is a similar measure for the optimality gap.

	Gurobi with initial solution				**CMBD**					
Case	**UB** (10^6\$)	**LB** (10^6\$)	**Gap** (%)	**Time** (min)	**UB** (10^6\$)	**LB** (10^6\$)	**Gap** (%)	**Time** (min)	**Speedup**	**Gap improv.**
1	60,573.21	60,308.39	0.4372	180	60,362.02	60,307.73	0.09	13	13.8	4.9
2	60,730.46	60,437.44	0.4825	180	60,492.76	60,436.75	0.093	13	13.8	5.2
3	60,232.31	60,231.73	0.001	102	60,283.84	60,230.85	0.088	13	7.8	0.0
4	60,619.68	60,290.37	0.5432	180	60,347.69	60,289.62	0.096	14	12.9	5.7
5	60,691.71	60,270.13	0.6946	180	60,325.56	60,269.48	0.093	12	15.0	7.5
6	57,076.53	56,840.65	0.4133	180	56,892.03	56,839.90	0.092	11	16.4	4.5
7	57,412.09	56,935.07	0.8309	180	56,987.10	56,934.18	0.093	14	12.9	8.9
8	57,284.93	57,044.32	0.42	180	57,096.84	57,043.36	0.094	11	16.4	4.5
9	57,368.64	57,170.37	0.3456	180	57,219.46	57,169.49	0.087	11	16.4	4.0
10	57,417.06	57,263.04	0.2683	180	57,309.11	57,262.04	0.082	10	18.0	3.3
11	121,346.49	120,386.48	0.7911	180	120,493.66	120,385.53	0.09	14	12.9	8.8
12	120,260.56	120,258.70	0.0015	178	120,373.36	120,257.88	0.096	10	17.8	0.0
13	121,971.17	120,337.35	1.3395	180	120,449.07	120,336.54	0.093	13	13.8	14.4
14	120,390.43	120,389.37	0.0009	175	120,500.74	120,388.54	0.093	16	10.9	0.0
15	122,195.66	120,376.12	1.489	180	120,482.81	120,375.50	0.089	10	18.0	16.7
16	83,350.65	82,905.68	0.5339	180	82,979.15	82,904.63	0.09	10	18.0	5.9
17	83,419.51	82,875.54	0.6521	180	82,947.85	82,874.42	0.089	10	18.0	7.3
18	83,404.96	82,928.44	0.5713	180	83,003.16	82,927.65	0.091	13	13.8	6.3
19	83,558.69	83,134.23	0.508	180	83,209.03	83,133.46	0.091	10	18.0	5.6
20	83,239.99	83,177.53	0.075	160	83,255.09	83,176.75	0.094	9	17.8	0.8
21	60,354.62	60,023.51	0.5486	180	60,075.98	60,022.57	0.089	11	16.4	6.2
22	60,331.78	60,065.63	0.4411	180	60,120.68	60,064.83	0.093	10	18.0	4.7
23	60,383.50	60,141.42	0.4009	180	60,197.41	60,140.58	0.094	12	15.0	4.3
24	60,420.42	60,202.67	0.3604	180	60,256.68	60,201.78	0.091	11	16.4	4.0
25	60,428.92	60,253.09	0.291	180	60,308.11	60,252.21	0.093	12	15.0	3.1

Table 9: Improving solutions, and the running time of each trial of Cases 14 and 20.

	Case 14		**Case 20**	
Trial	**Improving solutions**	**Time (min)**	**Improving solutions**	**Time (min)**
1	20	16	14	8.17
2	25	16.9	14	8.38
3	22	15.4	18	9.18
4	26	20.1	13	7.78
5	22	15.9	19	9.16
6	28	20.5	11	8.27
7	16	13.8	16	9.28
8	17	12.2	12	9.24
9	20	13.6	12	8.02
10	18	14.5	11	8.31

As one might expect, longer running times are associated with smaller improvements in the UB: if the newly found solutions do not significantly improve the UB, then more improving solutions have to be found, which, in turn, leads to longer times. This is exactly what Tab. 9 reflects for Case 14, trials 4 and 6 of this case take, respectively, 26 and 28 improving solutions, which cause their times to reach, respectively, 20.1 and 20.5 min. On the other hand, Case 14, Trial 7 only takes 16 improving solutions, driving the time down to 13.8 min. For Case 20, the same trend can be seen. However, in this case, the differences in the number of improving solutions are much smaller and that causes the differences in running times to be also small. The gaps, as seen by the General coordinator, at the moment that it receives the improving solutions are shown in Figures 7 and 8, respectively for Cases 14 and 20. In these figures, we can see that, for the trials in which convergence is achieved quickly, only a handful of improving solutions is necessary, and the decreases in gap are steep. In contrast, in trials that

take many improving solutions, the decreases in the gap are rather slow.

7. Conclusion and future work

In this paper, we have presented a Cooperative Multi-search Benders Decomposition (CMBD) specially designed for the hydrothermal unit-commitment problem (HTUC). The CMBD proposed in this work enables the simultaneous exploration of different regions of the master problem yielded by the BD. In addition to CMBD, we propose decoupling the updates of the upper bound and lower bound of the HTUC. This proposal is motivated by the tight dual bounds given by the continuous relaxation of the HTUC, as we have shown empirically in this work. We assess the proposed method with 25 test cases from a large-scale HTUC, and we use as benchmark the start-of-the-art optimization solver Gurobi. In all of our tests, CMBD far outperforms Gurobi. CMDB is, on average, 15 times faster than the optimization solver. Future works might include tuning of the several CMBD's parameters, inclusion of stochasticity and assessment of scalability.

References

[1] M. Tahanan, W. van Ackooij, A. Frangioni, F. Lacalandra, Large-scale unit commitment under uncertainty, 4OR 13 (2) (2015) 115–171. doi:10.1007/s10288-014-0279-y.

[2] M. Cordova, E. Finardi, F. Ribas, V. de Matos, M. Scuzziato, Performance evaluation and energy production optimization in the real-time operation of hydropower plants, Electric Power Systems Research 116 (2014) 201–207.

[3] Y. Wang, S. Wang, L. Wu, Distributed optimization approaches for emerging power systems operation: A review, Electric Power Systems Research 144 (2017) 127–135.

[4] M. Håberg, Fundamentals and recent developments in stochastic unit commitment, International Journal of Electrical Power & Energy Systems 109 (2019) 38–48.

[5] R. Taktak, C. D'Ambrosio, An overview on mathematical programming approaches for the deterministic unit commitment problem in hydro valleys, Energy Systems 8 (1) (2017) 57–79.

[6] P. Ramanan, M. Yildirim, E. Chow, N. Gebraeel, An asynchronous, decentralized solution framework for the large scale unit commitment problem, IEEE Transactions on Power Systems 34 (5) (2019) 3677–3686.

[7] M. J. Feizollahi, M. Costley, S. Ahmed, S. Grijalva, Large-scale decentralized unit commitment, International Journal of Electrical Power & Energy Systems 73 (2015) 97–106.

[8] Y. Wang, L. Wu, J. Li, A fully distributed asynchronous approach for multi-area coordinated network-constrained unit commitment, Optimization and Engineering 19 (2) (2018) 419–452.

[9] T. Santos, A. Diniz, C. Saboia, R. Cabral, L. Cerqueira, Hourly pricing and day-ahead dispatch setting in brazil: The dessem model, Electric Power Systems Research 189 (2020) 106709.

[10] Y. Chen, F. Wang, Y. Ma, Y. Yao, A distributed framework for solving and benchmarking security constrained unit commitment with warm start, IEEE Transactions on Power Systems 35 (1) (2019) 711–720.

[11] G. Iommazzo, C. D'Ambrosio, A. Frangioni, L. Liberti, A learning-based mathematical programming formulation for the automatic configuration of optimization solvers, in: International Conference on Machine Learning, Optimization, and Data Science, Springer, 2020, pp. 700–712.

[12] W. S. Sifuentes, A. Vargas, Hydrothermal scheduling using benders decomposition: accelerating techniques, IEEE Transactions on Power Systems 22 (3) (2007) 1351–1359.

[13] G. Byeon, P. Van Hentenryck, Unit commitment with gas network awareness, IEEE Transactions on Power Systems 35 (2) (2019) 1327–1339.

[14] L. Wu, An improved decomposition framework for accelerating lsf and bd based methods for network-constrained uc problems, IEEE Transactions on Power Systems 28 (4) (2013) 3977–3986.

[15] G. Schryen, Parallel computational optimization in operations research: A new integrative framework, literature review and research directions, European Journal of Operational Research.

[16] S. J. Maher, Implementing the branch-and-cut approach for a general purpose benders' decomposition framework, European Journal of Operational Research 290 (2) (2021) 479–498.

[17] C. K. Simoglou, P. N. Biskas, A. G. Bakirtzis, Optimal self-scheduling of a thermal producer in short-term electricity markets by milp, IEEE Transactions on Power Systems 25 (4) (2010) 1965–1977.

[18] R. M. Lima, A. Q. Novais, Symmetry breaking in milp formulations for unit commitment problems, Computers & Chemical Engineering 85 (2016) 162–176.

[19] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, Computational Management Science 2 (1) (2005) 3–19.

[20] R. Rahmaniani, T. G. Crainic, M. Gendreau, W. Rei, The benders decomposition algorithm: A literature review, European Journal of Operational Research 259 (3) (2017) 801–817.

[21] M. Fischetti, A. Lodi, Local branching, Mathematical programming 98 (1) (2003) 23–47.

[22] W. de Oliveira, C. Sagastizábal, Level bundle methods for oracles with on-demand accuracy, Optimization Methods and Software 29 (6) (2014) 1180–1209.

[23] W. de Oliveira, Regularized optimization methods for convex minlp problems, Top 24 (3) (2016) 665–692.

[24] O. N. do Sistema Elétrico, Mapas (2021).
URL http://www.ons.org.br/paginas/sobre-o-sin/mapas

[25] L. Gurobi Optimization, Gurobi optimizer reference manual (2021).
URL http://www.gurobi.com

[26] Lisandro Dalcin and Mikael Mortensen, mpi4py-fft.
URL https://bitbucket.org/mpi4py/mpi4py-fft

Figure 7: Gap at the moment the General coordinator receives an improving solution for Case 14. Different solutions might have the same gap in this figure because the LB given by the LB worker might not yet be available at the moment that the General coordinator receives the improving solution. Recall that $\log_{10}(100) = 2$ and $\log_{10}(0.1) = -1$.

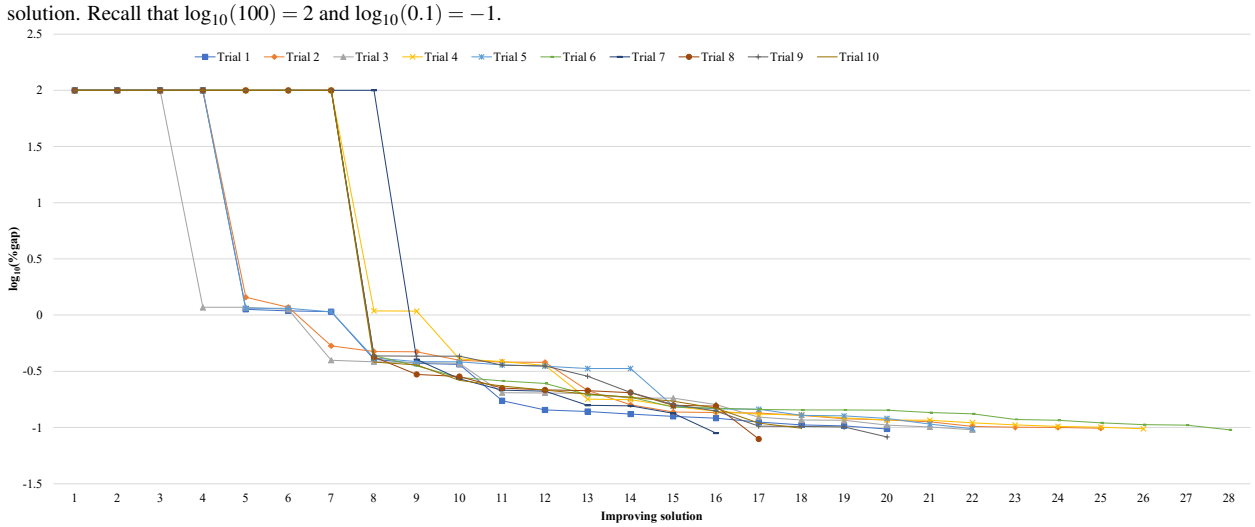


Figure 8: Gap at the moment the General coordinator receives an improving solution for Case 20. Different solutions might have the same gap in this figure because the LB given by the LB worker might not yet be available at the moment that the General coordinator receives the improving solution. Recall that $\log_{10}(100) = 2$ and $\log_{10}(0.1) = -1$.