Decomposition and Shortest Path Problem Formulation for solving the Hydro Unit Commitment and Scheduling in a Hydro Valley

Wim van Ackooij, Claudia d'Ambrosio, Dimitri Thomopulos, Renan Spencer Trindade

Decomposition and Shortest Path Problem Formulation for solving the Hydro Unit Commitment and Scheduling in a Hydro Valley

Wim van Ackooij Claudia D'Ambrosio Dimitri Thomopulos
Renan Spencer Trindade

1 Introduction

Managing an electrical system on a day to day basis involves a detailed modelling of the system as a whole. Traditionally, this management is achieved through unit-commitment, e.g., [1, 2]. In energy systems involving both thermal units and cascading systems, the latter do add quite some complexity to the overall resolution process. We refer to [3] for a detailed overview of cascading reservoir management. Centrally managing these resources is still vital today as carefully explained in [2]. It is however also naturally part of strategic planning, such as one would encounter, for instance investigating the efficient operation of the system 30 or more years from now. The European H2020-funded project plan4res explicitly builds these various layers of complexity of operation: investment, seasonal storage, unit-commitment operation.

Efficiently solving these unit-commitment models is thus important for practice. The specific nature of managing cascading systems immediately steps forth. First of all, the system naturally connects a variety of units, whether turbines or pumps, together through the overall cascading nature of the entire system. The natural connection is through the linear flow equations involving the release rates of each turbine (and pumping rates). The issue is however that the flow rate of a turbine is not necessarily nicely connected to the amount of power output. These so-called hydro-production functions are typically non-concave and a vast literature of approximations is available. We refer to [4, 5, 6, 7, 8, 9] and the references therein for more information. In some situations it may be convenient to actually discretize the hydro-production function, so as to consider the set of feasible flow rates to be completely discrete. There are some added advantages to this way of proceeding, and evidently some disadvantages. The main disadvantages are of course that the discrete variables naturally render the whole problem more rigid, especially if overall the flow equations and reservoir bounds are "tight". Moreover, it may well also be that a significant number of discrete variables is required to have a reasonably precise model. In terms of advantages, first of all the non-concave nature of the hydro-production function (possibly depending on the water head), is nicely linearized. Furthermore, it becomes possible to integrate with ease the amount of energy generated for "spinning reserve" requirements. This latter feature would otherwise require accounting for differences in the hydro-production function, not at all easy to model and handle. One can make this fact intuitively understandable already if the hydro production function is concave. Then representing the amount of energy generated by "spinning reserves" implies handling a difference of concave functions (at different points). Such a setting leads to a difference-of-convex constraint, of more difficult nature than the original constraint (e.g., [10]). In this paper, we have however opted for a full discretization of the hydro-production function.

Consequently, even with linearized "non-concave" functions, the model remains challenging to solve. This is especially true for the larger cascading systems. Moreover, we may add to this the necessity of solving the model quickly. The last requirement originates from two typical uses. First of all, unit-commitment itself is inserted in an operational process and typically not much time can be allocated to the optimization

itself. This may be, especially for complicated or large scale systems, a reason to move to decomposition based approaches. As a result, optimizing a given cascaded system is inserted into an iterative algorithm, and consequently performed several times. Note that the decomposition strategy also allows to exploit parallel computing of solutions of the different subproblems, speeding up the solution process. Second, when cascading reservoir management is considered to be part of strategic planning, the overall model is so large that decomposition must be employed, and once more optimization of the elementary building blocks must be extremely fast.

In this paper, we are thus concerned with optimization of a cascaded reservoir system facing a price signal (or Lagrange multiplier vector). The starting point is that either the monolithic model cannot be solved efficiently with state of the art commercial solvers, e.g., CPLEX, GUROBI, or that such solvers are not available for some reason. The second case can clearly be imagined if solving the cascading system is part of a local energy management system scheduler, or is deployed in some other solution wherein the savings do not outweigh the licensing costs. Our approach is based on three ingredients:

- The efficient resolution of a single reservoir, "single-unit" situation by casting the turbining / pumping constraints into a graph structure and using a constrained shortest path methodology, as introduced for the first time in this context by [11]
- The use of Lagrangian decomposition to uncouple the cascading structure
- The use of a bundle method to efficiently compute lower bounds and feasible schedules.

The paper is outlined as follows: in Section 2 we formally introduce the hydro unit commitment problem on hydro valleys. A formulation as shortest path problem and a solution method for the single-reservoir case is presented in Section 3. Then, the decomposition scheme and bundle method for solving the multiple reservoir case are introduced in Section 4. Finally, in Section 5 computational results show the effectiveness of the proposed approach.

2 The hydro unit commitment problem on hydro valleys

In this section, we formally introduce the deterministic Hydro Unit Commitment Problem (HUCP) on hydro valleys. We also call this problem Multiple Reservoir HUCP. We assume that the head-effect can be neglected (the produced power does not depend on the height from which the water falls from the reservoir to the downhill hydroplant) and, as mentioned in the previous section, we consider the operational points set as finite (discrete operational points). It is important to say that these operational points are used in practice by the management team. Therefore, this discretization reflects real practice, where the system is planned to operate with these previously defined points. Note that hydro plants might be composed of several turbine/pump units. Some of these turbines might be reversible and so evidently the turbining and pumping mode cannot be used simultaneously. Moreover, we also assume that there is an order in which the turbines have to be activated, e.g., in order to reduce cavitation and/or improve efficiency. In other words, the second unit is not turned on unless the first unit is functioning at full capacity, the third unit is not turned on unless the second unit is functioning at full capacity, and so on. Altogether, we order the different units in such a way, that they can be aggregated in a unique unit. The fully discretized nature of the problem makes this possible.

We start by introducing some notation (sets, parameters, variables), then a mathematical formulation.

We define the following sets:

- R: set of reservoirs. For ease of notation, we assume that, for each reservoir i, we are given a downhill hydro plant, thus R could also denote the set of hydro plants
- $J_i = \{1, \ldots, \bar{\jmath}_i\}$: set of operational points for hydro plant i, for all $i \in R$. In the rest of the paper, we assume that $\bar{\jmath}_i$ is the same for all $i \in R$ to simplify the notation. However, our approach can deal with different $\bar{\jmath}_i$ for all $i \in R$

- R_i^-: set of downstream reservoir for reservoir i, for all $i \in R$
- R_i^+: set of upstream reservoir for reservoir i, for all $i \in R$.

Note that J_i ($i \in R$) includes operational points both in turbine mode and in pump mode.
We use the following parameters:

- T: number of times steps in the considered time horizon
- Δ^T: time step length
- π_t: price for power at time step t, for all $t \in \{1, \dots, T\}$
- I_{it}: water inflow in reservoir i at time step t, for all $i \in R$ and $t \in \{1, \dots, T\}$
- Q_{ij}: water flow corresponding to operational point j of hydro plant i, for all $i \in R$ and $j \in J_i$
- P_{ij}: produced power corresponding to operational point j of hydro plant i, for all $i \in R$ and $j \in J_i$
- $\underline{\Phi}_i, \overline{\Phi}_i$: maximum water flow ramp-down/ramp-up between two consecutive time steps for hydro plant i, for all $i \in R$
- $\underline{V}_i$, $\overline{V}_i$: lower and upper bound on the water volume for reservoir i, for all $i \in R$
- $V_i^0 \in [\underline{V}_i, \overline{V}_i]$: initial water volume of reservoir i, for all $i \in R$
- $V_i^T \in [\underline{V}_i, \overline{V}_i]$: final target volume for reservoir i, for all $i \in R$.

Note that when $Q_{ij} > 0$ (resp. $Q_{ij} < 0$) then $P_{ij} > 0$ (resp. $P_{ij} < 0$), for $i \in R$ and $j \in J_i$. Moreover, if $Q_{ij} = 0$, then $P_{ij} = 0$ for $i \in R$ and $j \in J_i$. We assume, without loss of generality, that, for any $i \in R$, $Q_{i1} < Q_{i2} < \cdots < Q_{i(\bar{\jmath}-1)} < Q_{i\bar{\jmath}}$ and $P_{i1} < P_{i2} < \cdots < P_{i(\bar{\jmath}-1)} < P_{i\bar{\jmath}}$. Let us also define $j_i^0 \in J_i$, the index $j \in J_i$ for which $Q_{ij} = P_{ij} = 0$, for all $i \in R$.
Variables and simple bounds:

- $q_{it} \in [Q_{i1}, Q_{i\bar{\jmath}}]$: water flow for hydro plant i at time step t for all $i \in R$ and $t \in \{1, \dots, T\}$
- $z_{ijt} \in \{0, 1\}$: equal to 1 if operational point j is active for hydro plant i at time step t for all $t \in \{1, \dots, T\}, i \in R, j \in J_i$
- $v_{it} \in [\underline{V}_i, \overline{V}_i]$: water volume at reservoir i at time step t for all $i \in R$ and $t \in \{1, \dots, T\}$
- $p_{it} \in [P_{i1}, P_{i\bar{\jmath}}]$: produced (or consumed if negative) power for hydro plant i at time step t for all $i \in R$ and $t \in \{1, \dots, T\}$.

The constraints related to our problem are as follows:

$$q_{it} = \sum_{j \in J_i} Q_{ij} z_{ijt} \qquad \forall i \in R, t = 1, \dots, T \tag{1}$$

$$p_{it} = \sum_{j \in J_i} P_{ij} z_{ijt} \qquad \forall i \in R, t = 1, \dots, T \tag{2}$$

$$\sum_{j \in J_i} z_{ijt} = 1 \qquad \forall i \in R, t = 1, \dots, T \tag{3}$$

$$v_{it} = v_{i(t-1)} + \Delta^T I_{it} + \Delta^T \sum_{j \in R_i^+} q_{jt} - \Delta^T \sum_{j \in R_i^-} q_{jt} \qquad \forall i \in R, t = 1, \dots, T \tag{4}$$

$$-\underline{\Phi}_i \leq q_{it} - q_{i(t-1)} \leq \overline{\Phi}_i \qquad \forall i \in R, t = 1, \dots, T \tag{5}$$

$$v_{iT} \geq V_i^T \qquad \forall i \in R. \tag{6}$$

Variables v_{i0} are fixed to V_i^0 and q_{i0} is the water flow at hydro plant i at the beginning of the considered time horizon. Constraints (1) and (2) ensure that q and p are chosen in the discrete operational points set J_i and constraints (3) ensures that exactly one operational point for each hydro plant $i \in R$ and time step $t = 1, \ldots, T$ is active. Constraints (4) at each reservoir i and time step t represents the water volume conservation. Ramp-up/down bounds on the water flow of two consecutive time steps are considered in constraints (5), while, for each reservoir i, constraints (6) guarantee a minimum water volume V_i^T at the end of the time horizon.

Note that in (4) we assume, for ease of notation, that the turbining/pumping in the upstream/downstream reservoirs affects the reservoir itself instantaneously. However, the model and the following approach can be adapted to the more general case of problems involving flow delays.

Finally, a widely used objective function is to maximize the revenue given by selling the produced power p_{it} at market price π_t ($i \in R$, $t \in \{1, \ldots, T\}$) minus the cost of the bought power (when p_{it} is negative) and the startup cost is paid each time a turbine or pump is switched on.

It was shown in [12, 13] that problem (1)-(6) can be very challenging. In the sequel, we propose a method based on a decomposition framework combined with a shortest path based formulation. The latter approach is presented in the next section for the single reservoir case.

3 The shortest path problem formulation for the single reservoir case

Problem (1)-(6) might be impractical, especially for large or detailed instances. Therefore we propose a decomposition method, splitting the hydro valley problem into subproblems formulated as the Single Reservoir Hydro Unit Commitment Problem (HUCP-SR), one for each reservoir, where the volume of water depends also on the flow of the upstream reservoirs when present.

As a consequence, for every couple (a, b) of upstream reservoir a and downstream reservoir b of the original problem, we solve a HUCP-SR. As assumed, every turbine/pump has a unique uphill and downhill reservoir. This does not exclude however the possibility that several turbines connect various upstream reservoirs to any given reservoir. In terms of graph-language, each reservoir can have several ancestor reservoirs, but only one child reservoir. The way that the decomposition scheme will operate will ensure that, in every subproblem, the information about the downstream reservoir b is ignored. This follows from our decomposition scheme, since this reservoir is already part of the corresponding subproblem when reservoir b is considered an upstream reservoir itself.

It has already been shown in [11] how discrete HUCP-SR can be solved efficiently by formulating it as a graph problem. More precisely the problem is reduced to a (Resource) Constrained Shortest Path Problem ((R)CSPP) and solved through a labelling algorithm. Unlike the standard RCSPP, wherein resources are typically only increasing/decreasing over "time", the additional difficulty in our case is the immediate result of having to handle bilateral inequalities for each reservoir. Indeed, the volumes are to remain restricted within two bounds. This implies, in particular, that setting up appropriate dominance rules is not immediate. We now provide further details about this algorithm and its particularities.

3.1 Graph modelling

For each reservoir $i \in R$, we construct a graph $G_i = (N_i, A_i)$ (see, e.g., Figure 1), considering all the possible operational points at each time step t (for $t = 0, 1, 2, \ldots, T, T+1$), where N_i is the set of operational points at each time step and A_i is the set of possible arcs between nodes of N_i, which are only forward, and $T + 1$ is an additional fictional time step introduced in order to obtain a single destination. For ease of notation, in the following we do not consider the index of reservoirs in our formulation. As we assumed in Section 2, $\bar{\jmath}$ operational points are available at each time step. Therefore, the number of nodes is $|N| = \bar{\jmath}T + 2$, i.e., $\bar{\jmath}$ nodes for each time step and 2 artificial nodes, source s and

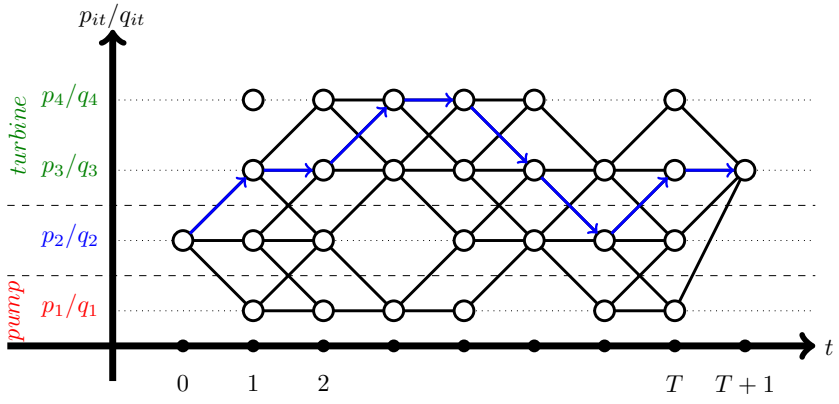

Figure 1: $G = (N, A)$

destination d, that represent time steps 0 and $T + 1$, respectively. They represent the initial and final node of the path corresponding to the operational profile of the unit. Graph G_i is a weighted single-source single-destination Directed Acyclic Graph (DAG).

Assuming $N_t = \{(t-1)\bar{\jmath}+1, (t-1)\bar{\jmath}+2, \ldots, t\bar{\jmath}\}$, the set of nodes that correspond to the operational points at time t, for $t \in \{1, \ldots, T\}$, then N can be defined as $\{s\} \cup \bigcup_{t=1}^{T} N_t \cup \{d\}$, where $\{s\} = \{0\}$ and $\{d\} = \{T\bar{\jmath}+1\}$. The arc $(i, j) \in A$ between nodes i and j exists if $i \in N_t$, $j \in N_{t+1}$, and if it is possible to move from the operational point corresponding to node i to the operational point corresponding to j without violating the physical constraints of the problem as ramp-up/down constraints (5).

In this fashion, all the constraints described in Section 2 referring to single reservoirs are considered in the feasibility of the arcs, except for the bounds on the water volume at each time step. Finally, the arcs are weighted with costs equal to the difference between the turbine/pump unit startup costs and the power selling revenue.

Thus, the deterministic HUCP-SR, under the assumptions previously mentioned, reduces to a Shortest Path Problem from s to d with extra bounding constraints on water volume of the reservoir, i.e., a (Resource) Constrained Shortest Path Problem. The objective of this problem is to minimize the cost coded in the arcs, all while ensuring feasible use of the resources, i.e., having valid water volumes for the reservoir. The problem can also be cast as an integer linear program, upon introducing binary variables x_{ij} $(\forall (i, j) \in A\ )$ representing

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is part of the selected path} \\ 0 & \text{otherwise.} \end{cases}$$

The volumetric constraints (resource constraints), then become:

$$\underline{V} \leq V^0 + \Delta^T \sum_{k=1}^{t} \left(I_k - \sum_{j \in N_k} \sum_{i:(i,j)\in A} Q_j x_{ij} \right) \leq \overline{V} \quad \forall t \in \{1, \ldots, T\}, \tag{7}$$

$$V^0 + \Delta^T \sum_{k=1}^{T} \left(I_k - \sum_{j \in N_k} \sum_{i:(i,j)\in A} Q_j x_{ij} \right) \geq V^T. \tag{8}$$

3.2 Monotone reformulation for labelling algorithm

As already mentioned, we propose a labelling algorithm in order to solve the HUCP-SR, which is one of the most applied state-of-the-art methods for solving the ((R)CSPP). The general labelling algorithm

consists in exploring all possible partial paths and storing in a pool for every node some labels containing information on the current resource values, depending on the previous chosen paths. For generating the new labels, all the labels in the previous nodes are considered, therefore the size of the pool might increase exponentially. It is evident that a crucial element of the algorithm is the reduction of all the possible labels, which is performed through a dominance step. Assuming that the resource functions are monotonically decreasing, a label l_1 is dominated by another label l_2 if each resource of l_1 is less than or equal to the value of the resource of l_2. When a label is dominated, it becomes unnecessary (it cannot be part of any optimal path) and it can be removed from the pool of available labels. The algorithm is therefore subject to strong assumptions of monotonicity of the resources. In case the monotonicity is not guaranteed, the dominance step in not applicable, making the algorithm inefficient. Indeed, the profile of the volume resource in HUCP-SR is not monotone, as the simultaneous presence of pump and turbines might produce a fluctuating behavior. During the time step where the chosen operational point belong to turbining mode, the volume is decreasing, while during the time step where the operational points belong to pumping mode, the volume is increasing. Therefore, it is necessary to apply some reformulations in order to set up a valid and useful dominance rule (see [11]).

We can show that constraints (7)-(8) can be rewritten as follows:

$$\Delta^T \sum_{k=1}^{t} \sum_{j \in N_k} \sum_{(i,j) \in A} \Delta Q_j x_{ij} \leq V^0 + \Delta^T \left(\sum_{k=1}^{t} I_k - t\ Q_0 \right) + \hat{V}_t \quad \forall t \in \mathcal{T} \tag{9}$$

$$\Delta^T \sum_{k=1}^{t} \sum_{j \in N_k} \sum_{(i,j) \in A} \Delta Q_j x_{ij} \geq V^0 + \Delta^T \left(\sum_{k=1}^{t} I_k - t\ Q_0 \right) - \overline{V}_t \quad \forall t \in \mathcal{T} \tag{10}$$

with

$$\hat{V}_t = \min \left(-V^T + \Delta^T \left(\sum_{k=t+1}^{T} I_k - (T-t)\, Q_0 \right), - \max_{\tau \in \mathcal{T}: \tau \leq t} (\underline{V}_\tau) \right) \quad \forall t \in \overline{\mathcal{T}}$$

$$\hat{V^T} = \min \left(-V^T, -\underline{V}_t \right),$$

where $\mathcal{T} = \{1, \ldots, T\}$, $\overline{\mathcal{T}} = \{1, \ldots, T-1\}$, and $\Delta Q_j = Q_j - Q_0$.

Assuming that at least one operational point of pumping mode exists, $Q_0 \leq 0$. If there is no such point, no reformulation is required.

The new fictional volume resource, modelled by constraints (9)-(10), is monotone additive and increasing. In addition also its bounds are monotonically increasing, as the target volume V^T is by definition greater than or equal to the original lower bound $\underline{V}$. Therefore, the dominance rule selects labels with the smallest quantity of resources used, i.e., cost and fictional volume. Because the right hand sides of constraints (10) are monotonically increasing, we can therefore propose a variant of the classical labelling algorithm (see Algorithm 1), where for every time step t, the dominance rule can be applied only if the following additional conditions are satisfied

$$V^0 + \Delta^T \sum_{k=1}^{t} \left(I_k - \sum_{j \in N_k} \sum_{i:(i,j) \in A} Q_j x_{ij} \right) - (T-t)\, Q_0 \leq \overline{V} \quad \forall t \in \mathcal{T}, \tag{11}$$

where, selecting always Q_0 in all remaining periods, the lower bound on the volume is satisfied.

In Algorithm 1, U is the set of partial paths to explore, while P is the set of partial paths already explored. l , l', and l^* are partial paths with attached labels storing the values of resources. We also introduce some notation in order to let the algorithm be more understandable. Assuming a to be a generic arc, v a generic node, and l a generic partial path, $(\sim, v)$ is every path ending in node v and (l, v) is a partial path ending in node v. The function $EXTEND(l, a)$ extends a partial path, given the path l and the arc a to add, modifying accordingly the resources of the labels. The function $FEASIBLE(w)$, given an

ending node w, verifies the feasibility of the arc, i.e., verifying the satisfaction of constraints (9)-(10). Finally, the function $REMOVE-DOMINATED(U,P)$ is the aforementioned dominance step, which removes all dominated labels from the pools U and P.

In addition, the labels in the pool are ordered applying a quick sort algorithm.

Algorithm 1: Labelling algorithm variant

Input : directed graph $G=(N,A)$ with
start node $s \in N$ and end node $d \in N$
resource windows for all nodes
resource vectors for all arcs

Output: pareto-optimal s-d path l^* with minimal cost

$U \leftarrow (\epsilon, s)$ and $P \leftarrow \emptyset$
forall $v \in N \backslash \{d\}$ **do**
  while $\exists l = (\sim, v) \in U$ **do**
    $U \leftarrow U \backslash \{l\}$
    forall $a = (v,w) \in A$ **do**
      $l' = (l,w) \leftarrow$ EXTEND (l,a)
      if $l' \in$ *FEASIBLE(w)* **then**
        $U \leftarrow U \cup l'$
      else
        discard l'
      end
    end
    $P \leftarrow P \cup l$
    if *Conditions (11) are satisfied* **then**
      $U, P \leftarrow$ REMOVE-DOMINATED (U,P)
    end
  end
end
$l^* \in P \mid$ cost$(l^*) ==$ min(cost$(l = (\sim, d) \in P)$)

4 Lagrangian approaches for the multiple reservoir hydro unit commitment

We now consider the case of two interconnected reservoirs and propose a method to decompose the problem, while having in mind the graph formulation presented in the previous section. Although this case serves as the base example, the suggested methodology can be applied to the general case, as shown in Section 5.

4.1 Decoupling the cascading structure through Lagrangian decomposition

Aiming to model a system with two reservoirs, we consider two different graphs: $G^1 = (N^1, A^1)$ and $G^2 = (N^2, A^2)$. Each graph represents a single reservoir structure consisting of several units that can function either as turbines or as pumps, as explained in Section 3. Variable q_t^1 represents the water flow leaving reservoir 1 and entering reservoir 2 in period t, and is the unique variable appearing in constraints for both reservoirs 1 and 2.

The resulting formulation is thus as follows:

$$(P_2) \qquad \min \sum_{(i,j)\in A^1} C^1_{ij}x^1_{ij} + \sum_{(i,j)\in A^2} C^2_{ij}x^2_{ij} \tag{12a}$$

$$\sum_{(j,i)\in A^1} x^1_{ji} - \sum_{(i,j)\in A^1} x^1_{ij} = \begin{cases} -1 & \text{if } i = 1; \\ 1 & \text{if } i = n_1; \\ 0 & \text{if } 1 < i < n_1. \end{cases} \tag{12b}$$

$$\sum_{(j,i)\in A^2} x^2_{ji} - \sum_{(i,j)\in A^2} x^2_{ij} = \begin{cases} -1 & \text{if } i = 1; \\ 1 & \text{if } i = n_2; \\ 0 & \text{if } 1 < i < n_2. \end{cases} \tag{12c}$$

$$q^1_t = \sum_{j\in N^1_t} \sum_{i:(i,j)\in A^1} Q^1_j x^1_{ij} \qquad \forall t \in \mathcal{T} \tag{12d}$$

$$q^2_t = \sum_{j\in N^2_t} \sum_{i:(i,j)\in A^2} Q^2_j x^2_{ij} \qquad \forall t \in \mathcal{T} \tag{12e}$$

$$v^1_t = v^1_{t-1} + \Delta_t(I^1_t - q^1_t) \qquad \forall t \in \mathcal{T} \tag{12f}$$

$$v^2_t = v^2_{t-1} + \Delta_t(I^2_t - q^2_t + q^1_t) \qquad \forall t \in \mathcal{T} \tag{12g}$$

$$v^1_t \geq V^T_1 \tag{12h}$$

$$v^2_t \geq V^T_2 \tag{12i}$$

$$\underline{V}^1 \leq v^1_t \leq \overline{V}^1 \qquad \forall t \in \mathcal{T} \tag{12j}$$

$$\underline{V}^2 \leq v^2_t \leq \overline{V}^2 \qquad \forall t \in \mathcal{T}. \tag{12k}$$

The objective function (12a) minimizes the cost of power generated or consumed in both reservoirs. Constraints (12b)-(12c) ensure the flow conservation in the graphs G^1 and G^2. Constraints (12d) model the water flow leaving (or entering) reservoir 1 and entering (or leaving) reservoir 2 in period t. Constraints (12e) model the water flow leaving reservoir 2 in period t. Constraints (12f)-(12g) model the water volume in the reservoir 1 and 2 in period t. Constraints (12h) and (12i) concern the final target volume for each of the reservoirs. Constraints (12j)-(12k) model the bounds on the water volume and ensure the minimum target water volume to be reached at the end of the time horizon.

If we consider each of the reservoirs separately, we can model the two problems as explained in Section 3. However, constraints (12g) link the two problems, which should be considered together.

Our idea is to decompose the problem into two subproblems, one related to each of the reservoirs, by relaxing the linking constraints.

To do so, we follow an age-old two step procedure: variable duplication and decomposition. The duplication step is particular in our case however. Indeed, we generate two copies of variables v^2_t. The first copy, denoted $\bar{v}^2_t$, considers the accumulated water flow value from reservoir 1 to reservoir 2 at time t. The second, denoted $\hat{v}^2_t$, considers the volume of water in reservoir 2, excluding water flow from reservoir 1. It is important to mention that the initial water volumes of the new variables $\bar{v}^2_0$ and $\hat{v}^2_0$ are set to be equal to V^1_0 and V^2_0, respectively. The constraints of the problem are essentially unaltered:

$$\text{(12b)} - \text{(12f)}$$

$$\bar{v}^2_t = \bar{v}^2_{t-1} + \Delta_t q^1_t \qquad \forall t \in \mathcal{T} \tag{13a}$$

$$\hat{v}^2_t = \hat{v}^2_{t-1} + \Delta_t(I^2_t - q^2_t) \qquad \forall t \in \mathcal{T} \tag{13b}$$

$$\underline{V}^2 \leq \bar{v}^2_t + \hat{v}^2_t - V^1_0 \leq \overline{V}^2 \qquad \forall t \in \mathcal{T} \tag{13c}$$

$$\bar{v}^2_0 = V^1_0 \tag{13d}$$

$$\hat{v}^2_0 = V^2_0 \tag{13e}$$

$$\text{(12h)} - \text{(12j)}.$$

The constraints (12g) and (12k) are replaced by the constraint (13c). The next step is to relax the new complicating constraint (13c) by dualizing it, i.e., by moving to the Lagrangian dual. The Lagrangian dual function attributes to the Lagrangian multipliers $(\lambda_t^{\mathtt{min}}, \lambda_t^{\mathtt{max}})$ the (optimal) value of the following problem:

$$
\begin{aligned}
(RL(\lambda)) \qquad \min \; & \sum_{(i,j)\in A^1} C_{ij}^1 x_{ij}^1 + \sum_{(i,j)\in A^2} C_{ij}^2 x_{ij}^2 + \\
& \sum_{t\in\mathcal{T}} \lambda_t^{\mathtt{min}} (\underline{V}^2 - \bar{v}_t^2 - \hat{v}_t^2 + V_0^2) + \\
& \sum_{t\in\mathcal{T}} \lambda_t^{\mathtt{max}} (-\overline{V}^2 + \bar{v}_t^2 + \hat{v}_t^2 - V_0^2) \qquad (14a) \\
& (12b) - (13d).
\end{aligned}
$$

It is an elementary observation that problem (14a) has a block structure: a first block considers only variables and constraints related to reservoir 1, whereas the second block has only variables and constraints related to reservoir 2. Solving problem (14a), thus amounts to solving two independent (sub)problems. For the sake of completeness, we provide the full formulation of these two subproblems. The first problem is

$$
\begin{aligned}
(SP_1(\lambda)) \qquad \min \; & \sum_{(i,j)\in A^1} C_{ij}^1 x_{ij}^1 + \sum_{t\in\mathcal{T}} \bar{v}_t^2 (\lambda_t^{\mathtt{max}} - \lambda_t^{\mathtt{min}}) && && (15a) \\
& \sum_{(j,i)\in A^1} x_{ji}^1 - \sum_{(i,j)\in A^1} x_{ij}^1 = \begin{cases} -1 & \text{if } i = 1; \\ 1 & \text{if } i = n_1; \\ 0 & \text{if } 1 < i < n_1. \end{cases} && && (15b) \\
& q_t^1 = \sum_{j\in N_t^1} \sum_{i:(i,j)\in A^1} Q_j^1 x_{ij}^1 && \forall t \in \mathcal{T} && (15c) \\
& v_t^1 = v_{t-1}^1 + \Delta_t (I_t^1 - q_t^1) && \forall t \in \mathcal{T} && (15d) \\
& \bar{v}_t^2 = \bar{v}_{t-1}^2 + \Delta_t q_t^1 && \forall t \in \mathcal{T} && (15e) \\
& \bar{v}_0^2 = V_0^1 && && (15f) \\
& \underline{V}^1 \le v_t^1 \le \overline{V}^1 && \forall t \in \mathcal{T} && (15g)
\end{aligned}
$$

and is equivalent to the HUCP-SR. We can solve this problem by using Algorithm 1 described in the previous section. It suffices to modify the objective function, which needs to consider the contribution of the relaxed constraint. The algorithm needs to calculate the $\bar{v}_t^2$ values for each iteration, which is difficult as long as the nodes that belong to each labeled solution are stored.

The second problem is

$$
\begin{aligned}
(SP_2(\lambda)) \qquad \min \; & \sum_{(i,j)\in A^2} C_{ij}^2 x_{ij}^2 + \sum_{t\in\mathcal{T}} \hat{v}_t^2 (\lambda_t^{\mathtt{max}} - \lambda_t^{\mathtt{min}}) && && (16a) \\
& \sum_{(j,i)\in A^2} x_{ji}^2 - \sum_{(i,j)\in A^2} x_{ij}^2 = \begin{cases} -1 & \text{if } i = 1; \\ 1 & \text{if } i = n_2; \\ 0 & \text{if } 1 < i < n_2. \end{cases} && && (16b) \\
& q_t^2 = \sum_{j\in N_t^2} \sum_{i:(i,j)\in A^2} Q_j^2 x_{ij}^2 && \forall t \in \mathcal{T} && (16c) \\
& \hat{v}_t^2 = \hat{v}_{t-1}^2 + \Delta_t (I_t^2 - q_t^2) && \forall t \in \mathcal{T} && (16d) \\
& \hat{v}_0^2 = V_0^2 && && (16e)
\end{aligned}
$$

and, in this case, we do not have the volume constraints, which were relaxed. This problem is a Shortest Path Problem (SPP), since we can rewrite the model to remove the variable $\hat{v}_t^2$ and the constraints (16c)-(16e), and use the objective function (17).

$$\min \sum_{(i,j)\in A^2} \left(C_{ij}^2 - \Delta_t \sum_{t\in\mathcal{T}: j\in N_t^2} \sum_{k=t}^{T} (\lambda_t^{\texttt{max}} - \lambda_t^{\texttt{min}}) Q_j^2 \right) x_{ij}^2 + \sum_{t\in\mathcal{T}} (\lambda_t^{\texttt{max}} - \lambda_t^{\texttt{min}}) \left(V_0^2 + \Delta_t \sum_{k=1}^{t} I_k^2 \right) \quad (17)$$

In both subproblems, the algorithms need to return the optimal objective function value and optimal path.

The Lagrangian dual problem consists of maximizing the Lagrangian dual function over all multiplier pairs $(\lambda_t^{\texttt{min}}, \lambda_t^{\texttt{max}}) \geq 0$. It is as follows:

$$\begin{aligned}
(DP) \qquad & \sup\ (SP_1(\lambda)) + (SP_2(\lambda)) + && (18a)\\
& \sum_{t\in\mathcal{T}} \lambda_t^{\texttt{min}} (\underline{V}^2 + V_0^2) + && (18b)\\
& \sum_{t\in\mathcal{T}} -\lambda_t^{\texttt{max}} (\overline{V}^2 + V_0^2) && (18c)\\
& \lambda_t^{\texttt{min}} \geq 0 \qquad\qquad t \in \mathcal{T} && (18d)\\
& \lambda_t^{\texttt{max}} \geq 0 \qquad\qquad t \in \mathcal{T}. && (18e)
\end{aligned}$$

As is well known in Lagrangian decomposition, e.g., [14, 15], one should not expect the corresponding solutions obtained while solving (18a)-(18e) to be feasible. This is not necessarily an issue however, since in many cases a good primal recovery scheme can be set up, see Section 4.3. Such schemes are particularly well developed in unit-commitment, e.g., [16] and in part strongly rely on re-using the information generated while solving (18a)-(18e).

4.2 Maximizing the Lagrangian dual

Maximizing the Lagrangian dual, is achieved by using a proximal bundle method ([17, 18, 19, 20, 21, 22]).

Bundle methods are state of the art for maximizing the Lagrangian dual (18a)-(18e), e.g., [23]. For technical details, we refer the interested reader to [21, 24]. We have opted for an aggregate bundle method, i.e., one that considers a single cutting plane model for the dual function. As a stopping condition, we have opted for a condition involving control both over the size of the aggregate linearization error and the norm of the aggregate subgradient. Indeed when both terms (or alternatively the predicted decrease) are sufficiently small (in fact zero), the optimal dual solution is reached. The advantage of a separate condition is that the aggregate linearization error naturally compares to the value of the objective function (in our case a monetary one), whereas the aggregate subgradient relates to the volume constraints. The stopping tolerances can thus naturally be related to what is to be considered small in both situations. The proximal parameter is updated according to the "poorman's Newton" formula, e.g., [25].

4.3 Primal recovery

Bundle methods use the solutions to the subproblems $SP_1(\lambda)$ and $SP_2(\lambda)$ to generate complete solutions in each iteration. There is no guarantee that this solution is even feasible, since the constraints to the downstream reservoir have been relaxed. Therefore, a heuristic to create a primal recovery is implemented in order to find a good quality solution with low computational effort. At the end of each iteration of the bundle methods, a list of activated arcs for each subproblem is created. After the bundle method converges, the heuristic creates a simplified MILP from P_2 model, with activated arcs only, i.e., with the arcs that were part of the optimal solution of a subproblem at some iteration of the bundle method.

Thus, the primal recovery solution combines all the decisions found by the subproblems. We expect the number of active arcs to be much lower than in the original problem.

Note that we cannot guarantee that this heuristic finds a feasible solution for any instance of the problem. However, as the numerical experiments of Section 5 show, in practice we were always able to find a feasible solution for our instances. A mechanism to discard arcs could be implemented in larger instances, but it was not necessary in our tests, since the time needed to run the primal recovery was relatively short.

5 Computational Results

We now present the computational results comparing IBM Ilog CPLEX v. 12.9 on the MILP formulation and the bundle method. At each iteration of the bundle method, each (R)CSPP and SPP are solved exactly. It would have been possible to solve the problem approximately by discretizing the label space. However, we decided to pick an exact variant. As we see in the detailed tables, the bundle method is in general faster than CPLEX. We generated instances thanks to the Hydro unit commitment Instances Generator (HIG) which can be found here `http://www.lix.polytechnique.fr/Labo/Dimitri.Thomopulos/libraries/HIG.html`. We considered three classes of instances:

- Two reservoirs instances: they serve mainly as proof-of-concept of our approach.
- Three reservoirs instances, with a downhill reservoir and two uphill reservoirs, both of linked to the downhill reservoir. We call this class of instances Y-shaped because of the shape the graph representing the hydro valley. Together with the two reservoirs instances, they represent the main bricks for more realistic instances.
- Six reservoirs instances: inspired by a real-world valley of EDF, this class presents two Y-shaped blocks linked together to two reservoirs blocks.

The considered month for generating inflows and prices is of April 2006, as in [26]. All the generated instances are available in the web page mentioned above. We set a time limit of 600 seconds. All the tests were performed on an Interl(R) Core i7-7700HQ CPU 2.8GHz and 16GB of RAM running on Windows 64bits.

In Tables 1, 2, and 3 we present, from the first to the last columns: the number of operational points considered, the number of time periods considered, the numbers of arcs in G, and, for each method, i.e., CPLEX and bundle based approach, the CPU time needed, the relative deviation from the best solution found for the upper bound (UB_d), and for the lower bound (LB_d). These values are calculated by $UB_d = 100(UB - UB^*)/UB^*$ and $LB_d = 100(LB^* - LB)/LB^*$, where LB^* and UP^* is the best known value for the lower bound and the upper bound, respectively. We also report the gap at the time limit and for the bundle method the number of iterations needed to converge.

We start with results for the 2 reservoirs case. In Table 1 we can clearly see that CPLEX is very effective when the number of operational points is limited, i.e., $\bar{j} \leq 7$. In these instances, the bundle approach is extremely fast. The approach finds the optimal solution in 3 cases out of 4. CPLEX also finds the optimal solution in these cases, but with more computational effort. In the fourth case, CPLEX can find a better solution than the bundle approach, but without closing the gap within the timelimit. As the number of operational points grows, the MILP becomes less tractable for CPLEX, which hits the time limit for all cases. CPLEX is strong in finding the best solution (UB) and the bundle approach finds the same solution for 4 instances out of 8. The latter approach never hits the limit, converging in 0.19 to 306 seconds, and finds the best LB for nearly all instances.

In Table 2, we present the results for the 3 reservoirs case. In the bundle approach, when the linking constraint is relaxed, we end up with a subproblems per reservoir: the ones corresponding to the two uphill reservoir are (R)CSPP problem, which can be solved in parallel, while the one corresponding to the downhill reservoir is a simple shortest path problem, solvable in polynomial time. For a fair comparison,

instances			CPLEX				Bundle				
$\bar{j}$	T	Arcs	time	UB_d	LB_d	gap	time	UB_d	LB_d	gap	it #
5	24	1170	55.92	**0.000**	**0.000**	**0.000**	0.190	**0.000**	0.549	0.549	8
5	48	2370	7.39	**0.000**	**0.000**	**0.000**	0.98	**0.000**	0.018	0.018	11
7	24	2282	600.00	**0.000**	**0.000**	**0.115**	0.40	0.189	0.236	0.542	11
7	48	4634	5.52	**0.000**	**0.000**	**0.000**	2.01	**0.000**	0.018	0.018	12
12	24	6672	600.00	**0.000**	0.155	0.344	5.23	0.022	**0.000**	**0.211**	10
12	48	13584	600.00	**0.000**	0.125	0.150	50.32	**0.000**	**0.000**	**0.025**	7
17	24	13362	600.00	**0.000**	0.226	0.302	4.34	0.001	**0.000**	**0.076**	9
17	48	27234	600.00	**0.000**	0.020	0.041	33.69	**0.000**	**0.000**	**0.021**	10
22	24	22352	600.00	**0.000**	0.353	**0.409**	21.39	1.095	**0.000**	1.164	8
22	48	45584	600.00	**0.000**	**0.000**	**0.050**	305.99	**0.000**	0.003	0.053	11
27	24	33642	600.00	**0.000**	0.138	**0.184**	11.63	0.867	**0.000**	0.920	9
27	48	68634	600.00	**0.000**	0.188	0.275	150.98	**0.000**	**0.000**	**0.087**	18

Table 1: Computational results for 2 reservoirs.

instances			CPLEX				Bundle				
$\bar{j}$	T	Arcs	time	UB_d	LB_d	gap	time	UB_d	LB_d	gap	it #
5	24	1755	262.33	**0.000**	**0.000**	**0.000**	0.220	**0.000**	0.005	0.005	10
5	48	3555	600.00	**0.000**	0.042	0.214	0.78	**0.000**	**0.000**	**0.172**	9
7	24	3423	600.00	**0.000**	1.246	1.368	0.29	**0.000**	**0.000**	**0.120**	8
7	48	6951	600.00	**0.000**	0.061	0.222	2.18	**0.000**	**0.000**	**0.161**	13
12	24	10008	600.00	**0.000**	0.299	0.479	5.67	0.112	**0.000**	**0.292**	9
12	48	20376	600.00	0.0000001	0.178	0.243	71.95	**0.000**	**0.000**	**0.065**	10
17	24	20043	600.00	**0.000**	0.276	0.302	4.55	0.094	**0.000**	**0.120**	8
17	48	40851	600.00	0.008	0.066	0.080	39.26	**0.000**	**0.000**	**0.006**	10
22	24	33528	600.00	**0.000**	0.460	0.484	30.61	0.108	**0.000**	**0.132**	10
22	48	68376	600.00	**0.000**	**0.000**	**0.014**	328.26	0.025	0.006	0.045	11
27	24	50463	600.00	0.225	0.196	0.488	15.00	**0.000**	**0.000**	**0.065**	10
27	48	102951	600.00	**0.000**	**0.000**	**0.030**	135.15	0.012	0.003	0.045	14

Table 2: Computational results for 3 reservoirs.

we run CPLEX with 2 threads as well. For the small Y-shaped instances, i.e., for $\bar{j} \leq 7$, CPLEX and the bundle approach find the same UBs. However, in this case, CPLEX could close the gap for the smallest instance only in 262.33 seconds. For the other 3 small instances, the bundle approach finds the best LB within less than 3 seconds. As for the larger instances, i.e., $\bar{j} \geq 12$, CPLEX finds the best UB for 5 out of 8 instances, while the bundle approach does so for 3 out of 8. However, the bundle approach provides the best LB for 6 instances out of 8, and CPLEX for the other 2, almost closing the gap within the timelimit of 600 seconds. The CPU time of the bundle approach varies between 4.55 to 329 seconds. CPLEX always hits the time limit, except for the smallest instance

In all the instances from both Tables 1 and 2, the bundle approach could find a feasible solution before applying the primal recovery. However, the primal recovery improved the solution found by the bundle approach twice on the 2 reservoirs instances and 5 times on the 3 reservoirs instances. The great advantage of the primal recovery is that its CPU time never exceeds 0.15 seconds in the first class of instances, 1.74 seconds in the second class.

Table 3 concerns the more realistic set of instances.

The decomposition creates 6 subproblems, two (R)CSPP and four SPP. We solved the (R)CSPP in parallel, while solving the SPPs is immediate. For a fair comparison, we run CPLEX with two threads. For this set of instances, CPLEX always hits the time limit, while the bundle approach takes from 2.23 seconds for the smallest instance and hits the limit for 4 instances out of 12. The bundle approach is superior to CPLEX for LB quality as well: it provides the best LB for 10 out of 12 instances. Comparing with the previous tables, it is clear that the bundle approach becomes more competitive concerning the LB as the number of reservoirs grows. This confirms the viability of the decomposition on realistic

instances			CPLEX				Bundle				
$\bar{j}$	T	Arcs	time	UB_d	LB_d	gap	time	UB_d	LB_d	gap	it #
5	24	3510	600.00	0.689	0.876	2.086	2.230	**0.000**	**0.000**	**0.502**	98
5	48	7110	600.02	**0.000**	0.037	0.294	66.16	0.011	**0.000**	**0.268**	579
7	24	6846	600.02	0.909	0.685	2.166	3.39	**0.000**	**0.000**	**0.548**	88
7	48	13902	600.03	**0.000**	0.015	0.199	19.39	0.010	**0.000**	**0.195**	98
12	24	20016	600.04	**0.000**	0.220	**0.386**	142.33	-	**0.000**	-	166
12	48	40752	600.05	**0.000**	0.077	**0.394**	600.00	-	**0.000**	-	55
17	24	40086	600.06	**0.000**	0.185	**0.355**	137.82	-	**0.000**	-	201
17	48	81702	600.09	**0.000**	0.034	**0.094**	400.31	-	**0.000**	-	80
22	24	67056	600.08	**0.000**	0.271	**0.447**	600.98	-	**0.000**	-	126
22	48	136752	600.15	**0.000**	**0.000**	**0.051**	600.00	-	0.774	-	14
27	24	100926	600.07	**0.000**	0.126	0.335	309.76	0.088	**0.000**	**0.297**	157
27	48	205902	600.26	**0.000**	**0.000**	**0.139**	600.00	-	0.000	-	56

Table 3: Computational results for 6 reservoirs.

instances.

However, the UB is more difficult to find with the primal recovery heuristic. It could find a feasible solution only for 5 out of 12 instances and only in two of these cases is the UB better than the one found by CPLEX. The number of arcs considered in the heuristic is between 1% and 15% of the total number of arcs. Thus, the heuristic is very fast but, for more challenging instances, it can not always find a feasible solution. In this particular set of instances, the problematic subproblems are likely to be the ones corresponding to downhill reservoirs. To confirm our intuition, we run two variants of the heuristics, each of which considering the full set of arcs for the subproblems corresponding to a reservoir in a low level of the valley. Adding the two heuristics to the bundle approach allowed us to find feasible solutions for all the instances, a better UB than CPLEX in 5 times out of 12. Clearly, the heuristic variants are not as efficient as the primal recovery: the CPU time goes from 0.95 seconds to 100 (the time limit we set in this case). This shows that the selection of the arcs to be considered is crucial for establishing a good trade off between efficiency and feasibility.

Conclusion

In this work, the hydro unit commitment problem on hydro valleys is addressed, and a general definition for Multiple Reservoir HUCP is presented. The approach proposed in this paper uses Lagrangian decomposition to uncouple the cascading structure from the model into two different subproblems. The first is (R)CSPP, and is solved by the procedure proposed in [11], and the second is a common SPP problem. A bundle method was used to find lower bounds and enriched with a primal recovery procedure to produce feasible solutions for the problem. The computational tests consider small instances with two and three cascading reservoir instances, and also six cascading reservoirs, which presents a more realistic reproduction of a valley. The results show that CPLEX presents good results using the complete model, without the decomposition, mainly to find feasible solutions. However, our approach can find lower bounds of better quality in most cases, and the primal recovery procedure can often find better (feasible) solutions than CPLEX. As a future direction, we think that the development of a sophisticated method to select the most promising arcs to be considered by the primal recovery would highly improve the produced upper bounds, i.e., feasible solutions.

Acknowledgements

This research benefited from the support of the FMJH Program PGMO and from the support of EDF. We thank an anonymous reviewer whose remarks helped us to significantly improve the paper.

References

[1] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale unit commitment under uncertainty: a literature survey. *4OR*, 13(2):115–171, 2015.

[2] W. van Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85, 2018.

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

[4] E.C. Finardi and E.L. Da Silva. Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming. *IEEE Transactions on Power Systems*, 21(2):835–844, 2006.

[5] E.C. Finardi, E.L. Da Silva, and C.A. Sagastizábal. Solving the unit commitment problem of hydropower plants via Lagrangian relaxation and sequential quadratic programming. *Computational & Applied Mathematics*, 24(3):317–341, 2005.

[6] E.C. Finardi, F. Y. K. Takigawa, and B. H. Brito. Assessing solution quality and computational performance in the hydro unit commitment problem considering different mathematical programming approaches. *Electric Power Systems Research*, 136:212–222, 2016.

[7] W. van Ackooij, E. C. Finardi, and G. Matiussi Ramalho. An exact solution method for the hydrothermal unit commitment under wind power uncertainty with joint probability constraints. *IEEE Transactions on Power Systems*, 33(6):6487–6500, 2018.

[8] J. Kong, H. I. Skejlbred, and O. B. Fosso. An overview on formulations and optimization methods for the unit-based short-term hydro scheduling problem. *Electrical Power Systems Research*, 178(1):1–14, 2020.

[9] A. Lodi, E. Malaguti, G. Nannicini, and D. Thomopulos. Nonlinear chance-constrained problems with applications to hydro scheduling. *Mathematical Programming*, page , 11 2019.

[10] W. de Oliveira. The abc of dc programming. *Set-Valued and Variational Analysis*, 28:679–706, 2020.

[11] Wim van Ackooij, Claudia D'Ambrosio, Leo Liberti, Raouia Taktak, Dimitri Thomopulos, and Sonia Toubaline. Shortest path problem variants for the hydro unit commitment problem. *Electronic Notes in Discrete Mathematics*, 69:309–316, 2018.

[12] Youcef Sahraoui, Pascale Bendotti, and Claudia D'Ambrosio. Real-world hydro-power unit-commitment: Dealing with numerical errors and feasibility issues. *Energy*, 184(C):91–104, 2019.

[13] Youcef Sahraoui. *Short-term hydropower production scheduling : feasibility and modeling. (Planification de la production hydroélectrique au court terme : faisabilité et modélisation)*. PhD thesis, University of Paris-Saclay, France, 2016.

[14] A. Daniildis and C. Lemaréchal. On a primal-proximal heuristic in discrete optimization. *Math. Programming Series A*, 104:105–128, 2005.

[15] L. Dubost, R. Gonzalez, and C. Lemaréchal. A primal-proximal heuristic applied to french unitcommitment problem. *Mathematical Programming*, 104(1):129–151, 2005.

[16] A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. *IEEE Transactions on Power Systems*, 18:313–323, 2003.

[17] C. Lemaréchal. An algorithm for minimizing convex functions. *Information Processing*, 1974:552–556, 1974.

[18] C. Lemaréchal. An extension of davidon methods to nondifferentiable problems. *Mathematical programming study*, 3:95–109, 1975.

[19] A. Frangioni. About Lagrangian Methods in Integer Optimization. *Annals of Operations Research*, 139(1):163–193, 2005.

[20] A. Frangioni. Standard Bundle Methods: Untrusted Models and Duality. *Preprint*, pages 1–51, 2018.

[21] W. de Oliveira, C. Sagastizábal, and C. Lemaréchal. Convex proximal bundle methods in depth: a unified analysis for inexact oracles. *Math. Prog. Series B*, 148:241–277, 2014.

[22] W. de Oliveira and M. Solodov. A doubly stabilized bundle method for nonsmooth convex optimization. *Mathematical Programming*, 156(1):125–159, 2016.

[23] C. Sagastizábal. Divide to conquer: Decomposition methods for energy optimization. *Mathematical Programming*, 134(1):187–222, 2012.

[24] W. van Ackooij and A. Frangioni. Incremental bundle methods using upper models. *SIAM Journal on Optimization*, 28(1):379–410, 2018.

[25] C. Lemaréchal and C. Sagastizábal. An approach to variable metric bundle methods. *Lecture Notes in Control and Information Science*, 197:144–162, 1994.

[26] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello. A MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124, 2008.