Profit Sharing Mechanisms in Multi-Owned Cascaded Hydrosystems

Pedro Borges* Claudia Sagastizábal† Mikhail Solodov‡ Leo Liberti § Claudia D'Ambrosio ¶

Abstract

We consider the optimal management of hydropower generated by a cascade of three interconnected reservoirs owned by different agents. In this setting, water availability at the downhill reservoirs depends on decisions taken by the agents upstream. This creates an opportunity for the hydroplant at the top to withhold water and take advantage of situations with higher selling prices, which makes the overall decisions of the agents deviate from what can be considered best for the cascade as whole. In order to mitigate the market power of the hydroplant uphill, we propose a mechanism to enforce some collaborative behavior among the agents. This is achieved by agents transferring upstream fractions of their profit in exchange for water released from the top. The corresponding mathematical models are trilevel deterministic and trilevel stochastic linear programming problems, with uncertainty in prices and streamflows (exogenous inflows). For the stochastic variants, analyzed both in two- and multi-stage formulations, we propose new solution methods, extending to the trilevel nested setting the well-known L-Shaped and Stochastic Dual Dynamic Programming methods. A successful implementation of the later depends on certain floating cuts, that represent symbolically Benders-like linearizations. Convergence properties are discussed for some of the procedures. Numerical experiments confirm the interest of the approach, because with the proposed mechanism the top owner as well as the downhill hydroplants earn more money than when acting in an individualistic manner.

Keywords Cascaded hydroplants, Trilevel Optimization, Nested Multistage Stochastic Problems

1 Introduction

Hydroelectricity is the most widely used renewable energy worldwide (60% in 2020, according to [IRE20]). In countries like Brazil, Canada, China and Norway hydrogeneration largely dominates the power mix. In France, hydropower is only second to nuclear energy, roughly representing 10% of the total generation. The value of hydropower for energy systems relies on several important features of the technology. To start with, hydroelectricity is renewable and storable, as potential power can be stored in the form of water in the reservoirs. Storage is one of the biggest challenges in the transition to sustainable electricity systems, and reservoirs offer volumes in storage capacity still unthinkable, at this time, in terms of batteries. According to the estimations [Int18], reservoirs represented 94% of the installed global energy storage capacity worldwide in 2018. In addition, triggering hydraulic generation is fast: it suffices to release water from the uphill reservoirs for the whole cascade to start generating.

All these properties clearly make hydroelectricity very attractive and, more importantly, extremely useful to counter some downsides of systems dealing with both flexible generation and flexible consumption, as customary nowadays. Hydropower is a crucial tool for network operators to succeed compensating uncertainty and intermittency of renewable energy sources with fluctuations of prices and demand, in a manner that guarantees the stability and safety of the electricity system.

*Instituto de Matemática Pura e Aplicada, Rio de Janeiro, RJ, Brazil (pborges@impa.br)
†IMECC/UNICAMP, Campinas, São Paulo, Brazil (sagastiz@unicamp.br).
‡Instituto de Matemática Pura e Aplicada, Rio de Janeiro, RJ, Brazil (solodov@impa.br).
§LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France (liberti@lix.polytechnique.fr).
¶LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France (dambrosio@lix.polytechnique.fr).

The issue of optimal management of hydro-dominated power systems has been vastly studied, considering short, mid, and long planning times; see [GMH10]. How to properly manage a cascaded hydrosystem, however, still remains a challenge. A large portion of the literature focuses on a price-taker situation in which hydroplant owners cannot influence spot prices; see, for example, the extensive review [TD17]. For a cascade of hydroplants belonging to different owners, we also adopt a price-taking setting (prices are given by scenarios, see Figure 5). Our contribution is to propose to manage the cascade in a manner that mitigates market power due to water withheld upstream. For a cascade like the one with three hydroplants represented in Figure 1, water availability of the middle and bottom reservoirs highly depends on decisions taken at the levels upstream. If the top hydroplant plays opportunistically and withholds water, downstream owners will earn less profit. In our numerical experiments we observed that an individualistic management of the cascade can reach extremes in which, for the top hydroplant to increase profit in 1%, the other levels must lose up to 10% of their profit. The assessment and mitigation of market power in decentralized hydro-thermal systems is discussed in [KLBP01]. A formulation solving an equilibrium problem with equilibrium constraints was proposed in [CA13]. Recently, a duopoly formulation of competing hydroplants in the same cascade is analyzed in [MM20]. While market power is clearly an issue for the deployment of decentralized hydro systems, proposals of mitigation strategies are rarely found in the literature, with the exception of the work on the Brazilian wholesale water market [Kel99], a market which, however, has not been implemented in practice.

The mechanism for market power mitigation we investigate in this work is informally described by the left-hand side arrows in Figure 1. Formally, we deal with a trilevel optimization problem with decisions intertwined in a complex pattern. For the optimization problem at a given level, variables of the uphill problem appear in right-hand side of the constraints, and downhill variables appear in the objective functions; see, for instance, the deterministic formulation in (2) below. While the former dependence is a natural consequence of the cascade, because water released upstream increases the reservoir volume downhill, the latter is due to the mitigation mechanism, with downhill levels paying to the upstream plants part of the profit they have that exceeds the individualistic profits.

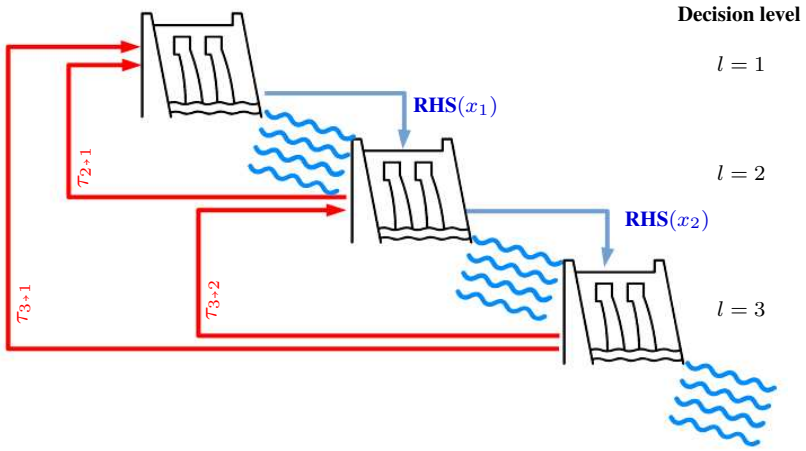

Figure 1: Water released uphill changes the right-hand side (RHS) in the water balance constraints of the power plant that is downhill. When water is withheld at some level, the profit of utilities that are downstream is diminished because they generate less power. To encourage water releases, downhill power plants transfer a fraction of their profit to utilities that are upstream. In the diagram, the utility in the top receives percentages $\tau_{2\to1}$ and $\tau_{3\to1}$ of the profit made by hydroplants in the middle and in the bottom, respectively. The latter also pays a fraction $\tau_{3\to2}$ of its profit to the middle power plant, that is immediately upstream. The arrows show how decisions are intertwined in the trilevel setting. Let x_l denote the decision taken when optimizing the generation of utility l. On the left, transferring profit upwards (left red arrows) goes from level $l+1$ to level l: the lth objective function includes terms involving downhill decisions x_{l+1} (for $l = 1$, the term includes both x_2 and x_3). On the right, the terms RHS(x_l) (right blue arrows) change the feasible set of the $(l+1)$th problem, and the dependence is reversed, going from level l to level $l+1$.

Figure 1 considers three independent hydrogenerators (top, middle, bottom) optimizing their profit according to some mathematical model. Their decisions are summarized by the generated power, a function of the reservoir volume, which in turns depends on the water released upstream. In our model, all relations are assumed to be linear, including the so-called efficiency function transforming water into energy (see [Ca+09; CaPM10] for nonlinear efficiency functions). Each generator aims at maximizing the profit resulting from selling the generated power at prices that can be random, while satisfying constraints, including bounds on the volumes. We consider a single objective function, for multi-objective formulations the reader is referred to [LQ18; CLY18]. Since streamflows to each reservoir are random, and prices increase if demand is high or if reservoir volumes are low, generators sometimes store water even in wet periods so that in dry periods, with higher prices, the released water generates power that yields more gains. In our proposal, agents in the middle and bottom levels transfer a fraction of their profit uphill, as an incentive to release water downstream. By this sharing mechanism, the profits of all the agents can only increase, when compared to the individualistic setting (the transfer is a fraction of what exceeds the individualistic profit). The numerical results reported in Figures 2 and 7 confirm that the top hydroplant as well as both downhill owners earn more money with our approach.

In Brazil, approximately 200 hydraulic utilities, distributed along 12 major river basins, belong to about 60 different companies; we refer to the map `www.ons.org.br/Mapas/Hidroel%C3%A9tricas%202022-2026%20Jan%202022.pdf` for full details.

The mathematical optimization problem has three levels of nested stochastic linear programs, for which we discuss variants with uncertainty unveiling in two-stage and multi-stage manners. All variables are continuous, to ensure convergence of the solution procedure described in Algorithm 1 below, where certain bilevel subproblems are reformulated by means of primal-dual relations that are necessary and sufficient optimality conditions in the considered setting. The different models and solution methods are given in Sections 2, 3, and 4. Section 2 presents the deterministic formulation of the trilevel model and introduces the proposed mechanism of transfer of profits. The solution approach, which defines cuts for the value functions of the successive levels as in a Benders' decomposition, is assessed on a cascade with three reservoirs like in Figure 1. We employ a simple, yet realistic, system that confirms the interest of transferring profit as a device to increase the gains of the cascade as a whole. The stochastic case is addressed in the next two sections. We first study, in Section 3, the individualistic model, which for the trilevel optimization problem amounts to having dependencies only on the feasible sets (in Figure 1 there are no arrows on the left, going upwards). We extend the L-shaped method [VW69] to a "cascaded" variant that properly deals with a sequence of three nested two-stage stochastic linear programs. To solve the more complex multi-stage model (still the individualistic formulation) we introduce a nested variant of the Stochastic Dual Dynamic Programming (SDDP) method [PP91] that deals simultaneously with the three levels. Section 3 ends with numerical results comparing the output of the individualistic model with a joint management of the cascade, both modeled in a multi-stage setting. The final Section 4 deals with the stochastic trilevel model with profit sharing. The resulting "cascaded" SDDP method is rather involved, its successful implementation relies on a new concept, that we named floating cut, described in Section 4.1. A floating cut is a symbolic representation of the Benders-like linearizations employed by the SDDP method. By this token, information is suitably transported between levels and the important properties of lower bounding and consistency are preserved throughout the iterative process. Regarding convergence, the methodology is shown in Theorem 2.1 to have finite termination in the deterministic case and also in the individualistic two-stage variant. For the remaining cases, our approach provides a reasonable heuristic whose good performance is confirmed by the numerical experiments carried on for this work. As an additional check of goodness, in the benchmark we compare in Subsection 4.3 the deterministic setting with the stochastic one, verifying that the output is consistent, with the respective results being similar to some extent.

2 Deterministic Trilevel Problem

It is convenient to cast the trilevel problem in an abstract format, that is particularized later on for the application. In order to gradually introduce the notation and setting, we consider first that there is no uncertainty and give the deterministic mathematical formulation, as well as a solution procedure for the resulting trilevel problem.

2.1 Problem Formulation

When considered independently, the optimization problem at level l is a linear programming problem of the form

$$\min_{x_l \geq 0} \quad f_l^\top x_l \quad \text{s.t.} \quad A_l x_l = a_l \,,$$

where the decision variable x_l has components such as turbined outflows and spillage, the lth reservoir volume, and the power generation. Water balance and capacity constraints are abstractly described by the matrix A_l and the vector a_l, of suitable dimensions. For the actual formulation see (13). Finally, the negative of the vector f_l in the objective function represents the unit profit made at level l.

To formalize the modifications due to the cascaded setting and formulate the trilevel problem, we start with the bottom level, $l = 3$. The effect that a release of water uphill (represented by x_2) has on the optimization problem downhill is measured by the following value function

$$v_3(x_2) := \min_{x_3 \geq 0} \quad f_3^\top x_3 \quad \text{s.t.} \quad A_3 x_3 = a_3 - B_3 x_2 \,.$$

As illustrated by the right arrows in Figure 1, changing a_3 to $a_3 - B_3 x_2$ relates levels $l-1$ and l through the right-hand side term (RHS).

Let us now define the problem for $l = 2$. The mitigation strategy proposes to transfer fractions $(\tau_{3\to 2}, \tau_{3\to 1})$ of the profit of generator of level 3 upstream, to levels 2 and 1, respectively. Regarding our level of interest, $l = 2$, this establishes the link in the objective function, relating levels l and $l+1$, represented by the left arrows in Figure 1. Specifically, because the term $-\tau_{3\to 2} v_3(x_2)$ represents a gain, it enters in the optimization problem with a negative sign. Accordingly, the modified objective function for level 2 is the cost

$$f_2^\top x_2 + \tau_{3\to 2} v_3(x_2)$$

and, hence, at level $l = 2$ the value function of interest is

$$v_2^{3\to 2}(x_1) := \min_{x_2 \geq 0} \quad f_2^\top x_2 + \tau_{3\to 2} v_3(x_2) \quad \text{s.t.} \quad A_2 x_2 = a_2 - B_2 x_1 \,.$$

A similar reasoning yields the objective function at the top level ($l = 1$):

$$f_1^\top x_1 + \tau_{2\to 1} v_2^{3\to 2}(x_1) + \tau_{3\to 1} v_3(x_2) \,, \tag{1}$$

where the decision variable x_2 is the decision that would be taken at $l = 2$ after x_1 is taken at $l = 1$.

In this nested optimization problem, the downhill value function $v_3(\cdot)$ is polyhedral and convex on x_2 and the function $v_2^{3\to 2}(\cdot)$ is convex on x_1 since $\tau_{3\to 2} \geq 0$, see [HUL93, Cor.IV.2.4.3]. Then, the function (1) is convex as a sum of convex functions whenever $\tau_{2\to 1}, \tau_{3\to 1} \geq 0$. Non-convexity arises in the problem because of the nexted formulation: the value of x_2 that enters (1) has to solve the optimization problem at level 2, as is made explicit in the formulation below by the notation x_2^{C}. For clarity, note that the value function $v_2^{3\to 2}(\cdot)$ is a function of $\tau_{3\to 2}$ as well as all the other problem data, which represents the costs and constraints. To make the notation more readable, we do not put in evidence these depencies, since we just exploit the properties of the value functions with respect to the decisions x_2 and x_3.

Summing up, in an optimistic setting, the trilevel problem amounts to finding

$$\begin{aligned}
x^{\mathsf{C}} = (x_1^{\mathsf{C}}, x_2^{\mathsf{C}}, x_3^{\mathsf{C}}) \text{ such that} \quad & x_1^{\mathsf{C}} \text{ solves } \begin{cases} \min\limits_{x_1 \geq 0} & f_1^\top x_1 + \tau_{2\to 1} v_2^{3\to 2}(x_1) + \tau_{3\to 1} v_3(x_2^{\mathsf{C}}) \\ \text{s.t.} & A_1 x_1 = a_1 \end{cases} \\
& x_2^{\mathsf{C}} \text{ solves } \begin{cases} \min\limits_{x_2 \geq 0} & f_2^\top x_2 + \tau_{3\to 2} v_3(x_2) \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^{\mathsf{C}} \end{cases} \\
& x_3^{\mathsf{C}} \text{ solves } \begin{cases} \min\limits_{x_3 \geq 0} & f_3^\top x_3 \\ \text{s.t.} & A_3 x_3 = a_3 - B_3 x_2^{\mathsf{C}} \,. \end{cases}
\end{aligned} \tag{2}$$

In our numerical experiments, we compare the output $x^{\mathtt{C}}$ of (2) with two other policies. First, a socially optimal policy that minimizes the aggregate total cost for the society, while being feasible for all agents:

$$\text{find } x^{\mathtt{S}} = (x_1^{\mathtt{S}}, x_2^{\mathtt{S}}, x_3^{\mathtt{S}}) \text{ solving} \quad \begin{cases} \min\limits_{(x_1,x_2,x_3)\geq 0} & f_1^\top x_1 + f_2^\top x_2 + f_3^\top x_3 \\ \text{s.t.} & A_1 x_1 = a_1 \\ & A_2 x_2 = a_2 - B_2 x_1 \\ & A_3 x_3 = a_3 - B_3 x_2 \,. \end{cases} \tag{3}$$

Second, individualistic policies that are optimally taken in a sequential manner along the cascade:

$$\text{find } x^{\mathtt{I}} = (x_1^{\mathtt{I}}, x_2^{\mathtt{I}}, x_3^{\mathtt{I}}) \text{ solving}$$

$$\begin{cases} \min\limits_{x_1\geq 0} & f_1^\top x_1 \\ \text{s.t.} & A_1 x_1 = a_1 \end{cases} \quad \begin{cases} \min\limits_{x_2\geq 0} & f_2^\top x_2 \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^{\mathtt{I}} \end{cases} \quad \begin{cases} \min\limits_{x_3\geq 0} & f_3^\top x_3 \\ \text{s.t.} & A_3 x_3 = a_3 - B_3 x_2^{\mathtt{I}} \,. \end{cases} \tag{4}$$

Since the RHS dependency may result in empty feasible sets, the policies might not be well defined in some configurations. We assume this is not the case, because feasibility can be ensured for our application by introducing slack variables representing energy deficits.

Notice that the individualistic point $(x_1^{\mathtt{I}}, x_2^{\mathtt{I}}, x_3^{\mathtt{I}})$ is feasible for the social problem (3). As a result,

$$v^{\mathtt{S}} := f_1^\top x_1^{\mathtt{S}} + f_2^\top x_2^{\mathtt{S}} + f_3^\top x_3^{\mathtt{S}} \leq v^{\mathtt{I}} := f_1^\top x_1^{\mathtt{I}} + f_2^\top x_2^{\mathtt{I}} + f_3^\top x_3^{\mathtt{I}} \,. \tag{5}$$

Policy makers are often concerned with the "social utility" of an energy market. The gap $v^{\mathtt{I}} - v^{\mathtt{S}} \geq 0$ gives a measure of the social dis-utility of the individualistic policy. We shall see that with our mechanism (2), of profit sharing, the social dis-utility of the market is decreased on the experiments we perform. Note that taking $\tau_{2\to 1} = \tau_{3\to 1} = \tau_{3\to 2} = 1$ is equivalent to transfering all the profits to level $l = 1$, which makes $l = 1$ select the socially optimal policy, since $l = 1$ receives all the profits and the socially optimal policy produces the largest profits. On the other hand, taking $\tau_{2\to 1} = \tau_{3\to 1} = \tau_{3\to 2} = 0$ induces the individualistic behavior. Therefore, if the social dis-utility behaves well with respect to $\tau_{2\to 1}, \tau_{3\to 1}$ and $\tau_{3\to 2}$, we expect that there are values for $\tau_{2\to 1}, \tau_{3\to 1}, \tau_{3\to 2} \in (0, 1)$ which induce a social dis-utility better than $v^{\mathtt{I}} - v^{\mathtt{S}}$. Nonetheless, we do not have a proof that the social dis-utility is a continuous function of $\tau_{2\to 1}, \tau_{3\to 1}$ and $\tau_{3\to 2}$, which is the ideal situation.

Naturally, we are interested in selecting fair values for $\tau_{2\to 1}, \tau_{3\to 1}$ and $\tau_{3\to 2}$. For such, we must have in mind that the socially optimal policy is the one that produces the largest profits, due to global coordination among the players. Transfering all profits to level 1 replicates the socially optimal profits, but clearly fails to improve the individualistic profits. Therefore, feasible values for $\tau_{2\to 1}, \tau_{3\to 1}, \tau_{3\to 2}$ should at least improve the individualistic profits. On the other hand, profits should be at least enough to pay the costs of running the hydro plant. These and other real-life considerations should determine "fair" values for $\tau_{2\to 1}, \tau_{3\to 1}$ and $\tau_{3\to 2}$.

2.2 Solution Procedure

To solve the trilevel problem (2) algorithmically we exploit the convexity of the value functions $v_2^{3\to 2}$ and v_3 in an iterative form. Letting k represent the current iteration, the respective cutting-plane approximations, denoted by v_2^k and v_3^k, are computed in a straightforward way. More precisely, for any given x_2^k, when solving the linear programming problem

$$v_3(x_2^k) = \min_{x_3\geq 0} \quad f_3^\top x_3 \quad \text{s.t.} \quad A_3 x_3 = a_3 - B_3 x_2^k \tag{6}$$

the equality constraints vector of multipliers λ_3^k is available. Convexity ensures the following inequality for the associated linearization:

$$\ell_3^k(x_2) := v_3(x_2^k) + (\lambda_3^k)^\top B_3 (x_2 - x_2^k) \leq v_3(x_2) \,.$$

The cutting-plane approximation for v_3 is then

$$v_3^k(x_2) := \max_{j=1,\dots,k} \ell_3^j(x_2) \leq v_3(x_2)\,.$$

Having this piecewise affine function, an analogous mechanism can be put in place for defining a cutting-plane model function that bounds $v_2^{3\to 2}$ from below. Since at level $l = 2$ the objective function involves the (unknown) value function v_3, linearizations are computed for an approximate function v_2^k, obtained when replacing v_3 by its cutting-plane model v_3^k:

$$v_2^k(x_1) := \min_{r_2, x_2 \geq 0} \quad f_2^\top x_2 + \tau_{3\to 2} r_2 \quad \text{s.t.} \quad A_2 x_2 = a_2 - B_2 x_1\,, \quad r_2 \geq \ell_3^j(x_2)\,, j = 1, \dots, k\,. \tag{7}$$

Solving this linear program with $x_1 = x_1^k$ gives a vector of multipliers λ_2^k for the equality constraints and, therefore,

$$\ell_2^k(x_1) := v_2^k(x_1^k) + (\lambda_2^k)^\top B_2(x_1 - x_1^k) \leq v_2^k(x_1)\,.$$

The linearization also bounds the function $v_2^{3\to 2}$ from below, because, by construction, the additional scalar variable in (7) satisfies $r_2 = v_3^k(x_2) \leq v_3(x_2)$, which implies that $v_2^k(x_1) \leq v_2^{3\to 2}(x_1)$ for all x_1, as claimed. Notwithstanding, it is important to keep in mind that the objective function in (7) is different from the one written for $l = 2$ in the trilevel problem (2). Specifically, Algorithm 1 replaces

$$f_2^\top x_2 + \tau_{3\to 2} v_3(x_2) \text{ from (2) by } f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2)\,, \tag{8}$$

using the function in (7). Such a replacement, necessary for the numerical implementation, has an impact on the convergence properties of the procedure; see Theorem 2.1 and the nearby comments.

In our procedure, the cutting-plane estimations of the value functions approximate the trilevel setting (2) by a sequence of bilevel linear problems

$$\text{Given } v_2^k\,, v_3^k, \text{ find } (x_1^k, x_2^k) \text{ such that} \quad x_1^k \text{ solves } \begin{cases} \min\limits_{x_1 \geq 0} & f_1^\top x_1 + \tau_{2\to 1} v_2^k(x_1) + \tau_{3\to 1} v_3^k(x_2^k) \\ \text{s.t.} & A_1 x_1 = a_1 \end{cases}$$
$$x_2^k \text{ solves } \begin{cases} \min\limits_{x_2 \geq 0} & f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2) \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^k\,. \end{cases} \tag{9}$$

These bilevel linear problems are solved as follows. First, the lower level problem is replaced by its equivalent optimality conditions, involving primal and dual feasibility and strong duality. The latter equality constraint introduces bilinear terms that are reformulated along the lines of McCormick cuts, using binary variables and a "big M" approach, i.e., choosing large constants that can make the problem ill-conditioned, due to bad scaling [Dem02, Ch. 5]. The resulting reformulation is a mixed integer linear programming problem. It is well known that success in the reformulation is driven by a sound choice of the aforementioned large constants.

Algorithm 1 gives the resulting iterative process in full detail.

Some comments regarding the different steps in Algorithm 1 are in order. Since we are dealing with polyhedral value functions, they are defined by a finite number of cutting-planes; in particular, by a finite number of subgradient values. The latter are the optimal Lagrange multipliers associated to the equality constraints in the linear programs (6) and (7). While the primal-dual solutions of these problems may not be unique in general, many (if not most) LP solvers compute *specific* solutions, which makes their selection finite. Moreover, these solutions correspond precisely to the cutting-planes whose maximum defines the polyhedral value functions exactly. For example, if the LP solver employed by Algorithm 1 computes basic optimal solutions (vertices), as the simplex method does, the number of optimal multipliers that the solver can return for (6) and (7) can only be finite; see, e.g., [OSS11, Prop. 4.1] (somewhat related considerations in a different context can also be found in [DSS09, p. 287]). Interior point methods which compute certain (unique) centered duals, see [ADCM91], also can return only a finite number of optimal solutions. The property of the LP solver producing a finite number of solutions, ensures finite termination of our algorithm; this can be seen, for example, from the argument in [Ber95, Prop. 6.3.2] for the cutting-plane method applied to a max-function.

Algorithm 1 SOLUTION PROCEDURE FOR TRILEVEL PROBLEM WITH SHARING MECHANISM. DETERMINISTIC CASE

Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$.
Set $k = 1$, $v_2^k(\cdot) \equiv -M$ and $v_3^k(\cdot) \equiv -M$.
Iterates for levels 1 and 2. Compute (x_1^k, x_2^k) solving the bilevel linear problem (9).
Iterate and linearization for level 3. Compute the primal-dual solution (x_3^k, λ_3^k) to the linear programming problem (6) and the linearization

$$\ell_3^{k+1}(x_2) = v_3(x_2^k) + (\lambda_3^k)^\top B_3(x_2 - x_2^k)\,. \tag{10}$$

Let $v_3^{k+1}(x_2) = \max\{v_3^k(x_2), \ell_3^{k+1}(x_2)\}$.
Linearization for level 2. Compute the primal-dual solution $(\hat{x}_2^k, \hat{r}_2^k, \lambda_2^k)$ to the linear programming problem

$$v_2^{k+1}(x_1^k) := \min_{r_2, x_2 \geq 0} \quad f_2^\top x_2 + \tau_{3\to 2} r_2 \quad \text{s.t.} \quad A_2 x_2 = a_2 - B_2 x_1^k\,, \quad r_2 \geq \ell_3^j(x_2)\,, j = 1, \ldots, k+1\,.$$

and $\ell_2^{k+1}(x_1) = v_2^{k+1}(x_1^k) + (\lambda_2^k)^\top B_2(x_1 - x_1^k)$. Let $v_2^{k+1}(x_1) = \max\{v_2^k(x_1), \ell_2^{k+1}(x_1)\}$.
Stopping test and loop. Stop if both the gaps of levels 2 and 3 are small, e.g., if

$$v_3(x_2^k) - v_3^k(x_2^k) \leq \varepsilon \quad \text{and} \quad v_2^{k+1}(x_1^k) - v_2^k(x_1^k) \leq \varepsilon.$$

Otherwise, set $k = k + 1$ and go back to **Iterates for levels 1 and 2**. □

Notwithstanding, a word of caution is in order, regarding the final iterate computed by Algorithm 1: it may not solve the trilevel problem (2), because the replacement (8) does not ensure the equality $v_3^k(x_2) = v_3(x_2)$ for all x_2. However, in practice, the output of Algorithm 1 provides at least a good estimate for practical purposes, as the inequality $v_3(x_2) \geq v_3^k(x_2)$ does hold everywhere. The next statement summarizes the convergence properties of Algorithm 1. The conditions needed for items (ii) and (iii) of Theorem 2.1 are discussed after the proof.

Theorem 2.1 (Finite termination of Algorithm 1)**.** *Consider Algorithm 1 with $\varepsilon = 0$, and let the linear programming solver therein be such that applied to* (6) *and* (7)*, only a finite number of primal-dual solutions can be returned by the solver. Then the following holds.*

(i) The algorithm terminates at some iteration k satisfying

$$v_3(x_2^k) = v_3^k(x_2^k) \quad \textit{and} \quad v_2^{k+1}(x_1^k) = v_2^k(x_1^k).$$

(ii) If $\tau_{3\to 1} = 0$, the iterate $x^k := (x_1^k, x_2^k, x_3^k)$ solves (2)*.*

(iii) If $v_3^k(x_2) = v_3(x_2)$ for all $x_2 \geq 0$ such that $A_2 x_2 = a_2 - B_2 x_1$ for all x_1, then x^k solves (2)*.*

Proof. (i) At some iteration k, the primal-dual points (x_2^k, λ_2^k) and (x_3^k, λ_3^k) eventually coincide with some other pairs computed previously, because the value functions are polyhedral and we assume that the linear solver produces only finitely many pairs of primal and dual solutions. By construction, the inequalities $v_3^{k+1}(x_2^k) \geq v_3^k(x_2^k)$ and $v_2^{k+1}(x_1^k) \geq v_2^k(x_1^k)$ hold. Since the point already defined linearizations at some past iteration, at the end of the kth iteration, by definition of v_3^k and v_2^k, we would have that $v_3^k(x_2^k) \geq v_3^{k+1}(x_2^k)$ and, analogously, $v_2^k(x_1^k) \geq v_2^{k+1}(x_1^k)$. The conclusion follows.

(ii) If $\tau_{3\to 1} = 0$, the problematic constraint on x_2 in problem (9) can be disregarded because we are assuming solvability. Then, convergence follows from standard properties of cutting-plane iterations.

(iii) When the algorithm stops, if $v_3^k(x_2) = v_3(x_2)$ for all x_2, then problem (9) represents perfectly problem (2). Then, x^k solves (2). □

Note that item (iii) of Theorem 2.1 refers to a sufficient condition for global optimality. Since problem (2) is not classical (nested, non-smooth) it is hard to develop necessary conditions based on derivatives of the functions involved, which may not even exist. In the sequel, we give a procedure to check the violation of the condition on item (iii).

As a result, when Algorithm 1 stops, its output x^k is feasible for the original problem (2), and the pair (x_1^k, x_2^k) solves globally the approximate problem

$$\text{Given } v_3^k, \text{ find } (x_1^k, x_2^k) \text{ such that} \quad x_1^k \text{ solves } \begin{cases} \min\limits_{x_1 \geq 0} & f_1^\top x_1 + \tau_{2\to 1} v_2^{3\to 2}(x_1) + \tau_{3\to 1} v_3^k(x_2^k) \\ \text{s.t.} & A_1 x_1 = a_1 \end{cases}$$

$$x_2^k \text{ solves } \begin{cases} \min\limits_{x_2 \geq 0} & f_2^\top x_2 + \tau_{3\to 2} v_3^k(x_2) \\ \text{s.t.} & A_2 x_2 = a_2 - B_2 x_1^k \,. \end{cases}$$

In view of these properties, Algorithm 1 can be thought of as being a Phase I procedure. If desired, a subsequent Phase II mechanism can be put in place to generate cuts that may be missing, therefore guaranteeing that $v_3^k(x_2) = v_3(x_2)$ for all x_2. By Theorem 2.1(iii), this triggers optimality of x^k for (2).

For other works analyzing the finiteness condition on the primal and dual solutions used on statement of Theorem 2.1, please see [Sha11; GB21; GR12].

Note that the condition $\tau_{3\to 1} = 0$ in item (ii) of Theorem 2.1 is not restrictive, because it can be enforced by the user of the model. In fact, $\tau_{3\to 1} = 0$ gives results with better social dis-utility in the computational experiments below; see Figure 2 in the sequel.

The condition in item (iii) is somewhat more restrictive. It requires that we have the full knowledge of the value function v_3, while in practice, we would have only a partial knowledge of it built using all previous iterates of Algorithm 1. To understand how the full knowledge of v_3 can be achieved, let us consider the problem

$$\max_{x_2} v_3(x_2) - v_3^k(x_2), \tag{11}$$

which finds a point x_2 where the mismatch between v_3 and v_3^k is maximized. Recall that $v_3(x_2) - v_3^k(x_2) \geq 0$. The issue with solving problem (11) is that it is non-convex and the function v_3 is not explicit. However, this problem can be solved quite easily by replacing (11) with

$$\min_{x_2, x_3^*} \quad v_3^k(x_2) - f_3^\top x_3^* \quad \text{s.t.} \quad x_3^* \in \arg\min\{f_3^\top x_3 \quad \text{s.t.} \quad A_3 x_3 = a_3 - B_3 x_2, \quad x_3 \geq 0\}. \tag{12}$$

Problem (12) can be solved using KKT reformulations and a MIP solver. Note also that it must be solved iteratively until the optimal value of (12) is zero.

2.3 Numerical assessment

The experiments are performed for the cascade of three hydropower plants represented in Figure 1 in the introduction. Over an horizon with $t = 1, \ldots, \mathtt{T}$ time periods, and for each hydroplant l,

- the data is: the efficiency factor $\boldsymbol{\Phi}_l$, the price of energy at time t denoted by $\boldsymbol{\Pi}^t$, the exogenous water inflow $\boldsymbol{A}_l^t$, the maximum and minimum reservoir volumes $\overline{\boldsymbol{V}}_l$ and $\underline{\boldsymbol{V}}_l$, and the maximum turbine outflow $\overline{\boldsymbol{U}}_l$.
- The variables are: the turbined outflow u_l^t, the reservoir volume v_l^t, and the spillage w_l^t.

The optimization problem solved by the hydropower plant l is shown below, where decisions of other levels are shown in bold and not present if $l - 1 < 1$:

$$\begin{cases} \max\limits_{(u,v,w)\geq 0} & \boldsymbol{\Phi}_l \sum\limits_{t=1}^{\mathtt{T}} \boldsymbol{\Pi}^t u_l^t & \\ \text{s.t.} & v_l^{t+1} = v_l^t + \eta(-u_l^t - w_l^t + \boldsymbol{A}_l^t + \boldsymbol{u}_{l-1}^t + \boldsymbol{w}_{l-1}^t), & t = 0, \ldots, \mathtt{T}-1 \\ & \eta(u_l^{\mathtt{T}} + w_l^{\mathtt{T}}) \leq v_l^{\mathtt{T}} + \eta \boldsymbol{A}_l^{\mathtt{T}}, & t = 1, \ldots, \mathtt{T} \\ & \eta u_l^t \leq v_l^t - \underline{\boldsymbol{V}}_l, & t = 1, \ldots, \mathtt{T} \\ & v_l^t \leq \overline{\boldsymbol{V}}_l, \quad u_l^t \leq \overline{\boldsymbol{U}}_l \,, & t = 1, \ldots, \mathtt{T}, \end{cases} \tag{13}$$

where η is the amount of unit time per time period. The meaning of constraints in (13) is standard, starting with the water balance in the reservoir, and inequalities to keep the outflow (turbined and spilt) within the capacity of the reservoir. Nonnegativity for the turbined outflow u_l^t rules out any pumping mechanism for simplicity (negative values could be handled as well).

As stated, problem (13) suffers from the end-of-period effect, that depletes reservoirs to maximize the profit of each agent. A final target volume could be incorporated to address this issue, but here we do not include that constraint, and focus on the stylized model (13). Additionally, to guarantee feasibility without resorting to deficit-related slack variables, neither the outflow nor the spillage are bounded above in (13). Finally, with respect to the abstract notation, we have the relations

$$x_l := \left((u_l^t, v_l^t, w_l^t, \text{slacks}_l^t)_{t=1}^{\mathtt{T}}\right), f_l := -\mathbf{\Phi}_l\left((\mathbf{\Pi}^t, 0, 0, 0)_{t=1}^{\mathtt{T}}\right),$$

where the slack variables are used to rewrite the feasible set in standard linear programming form, with equality constraints only.

Note that, for our specific application, the formulation is always feasible. First, if there is too much water, we just activate the spillage variables, which are not bounded from above and guarantee that the maximum volume stored is respected. Second, if the reservoirs start with more volume than the minimum, they will be maintained with volume greater than the minimum. Therefore, the only feasibility condition is that the initial configuration of the reservoirs is feasible.

The data defining our toy problem, with a simplified cascade configuration, is meant to illustrate the features of interest, but should not be considered a realistic system. Prices and inflows are the mean of those considered in the stochastic setting and shown in Figure 5. The McCormick-like reformulation of the bilevel problems (9) uses values for "big M" in $[10^4, 10^5]$, tuned numerically. The remaining parameters are given in Table 1.

Table 1: Data for deterministic runs

l	$\mathbf{\Phi}_l$	$\boldsymbol{V}_0$	$\underline{\boldsymbol{V}}_l$	$\overline{\boldsymbol{V}}_l$	$\overline{\boldsymbol{U}}_l$
1	1	0.24	0.16	1.60	0.80
2	1	0.15	0.10	1.00	0.50
3	1	0.24	0.16	1.60	0.80

The benchmark compares the output $x^{\mathtt{C}}$ of the profit-sharing mechanism (2) with the social and individualistic policies, $x^{\mathtt{S}}$ and $x^{\mathtt{I}}$ solving (3) and (4), respectively. The considered percentages for (2) are

$$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} \in \{0.02, 0.05, 0.10, 0.20, 0.30, 0.40\},$$

and additionally, for some other benchmarks, we fix $\tau_{3\to1} = 0.00$, as in Theorem 2.1(ii).

All experiments were run on an Intel i7 1.90GHz machine, with Ubuntu 18.04.3 LTS, Julia 1.1.1[Bez+17] and CPLEX 12.10. Solving each deterministic variant took less than 20 seconds for $\varepsilon = 10^{-4}$.

The profit of each variant is then compared to that obtained with the social policy. Since the latter is $-f_l^\top x_l^{\mathtt{S}}$, the difference in profit with the individualistic policy is $-f_l^\top(x_l^{\mathtt{S}} - x_l^{\mathtt{I}})$ for $l = 1, 2, 3$. With the profit sharing mechanism, the difference in profit is

$$\begin{cases} -f_3^\top x_3^{\mathtt{S}} + (1 - \tau_{3\to1} - \tau_{3\to2}) f_3^\top x_3^{\mathtt{C}} & \text{if } l = 3 \\ -f_2^\top x_2^{\mathtt{S}} + (1 - \tau_{2\to1})(f_2^\top x_2^{\mathtt{C}} + \tau_{3\to2} f_3^\top x_3^{\mathtt{C}}) & \text{if } l = 2 \\ -f_1^\top x_1^{\mathtt{S}} + f_1^\top x_1^{\mathtt{C}} + \tau_{2\to1}(f_2^\top x_2^{\mathtt{C}} + \tau_{3\to2} f_3^\top x_3^{\mathtt{C}}) + \tau_{3\to1} f_3^\top x_3^{\mathtt{C}} & \text{if } l = 1\,. \end{cases}$$

The individualistic profits $v^{\mathtt{I}}$ defined by (5) are useful to improve the quality of the output of the profit-sharing model. This is made clear by realizing that the policy remains unchanged if, for instance,

$$\text{at level } l = 2, \text{ the transfer is } \tau_{3\to2}(v_3^{\mathtt{C}}(x_2) - \phi_l) \text{ for any benchmark value } \phi_l. \tag{14}$$

We exploit this degree of freedom to improve the quality of the results in the profit-sharing model. Specifically in our numerical experiments, we take $\phi_l = v_{l+1}^{\mathtt{I}}$, so that rewards that go uphill are only a fraction of

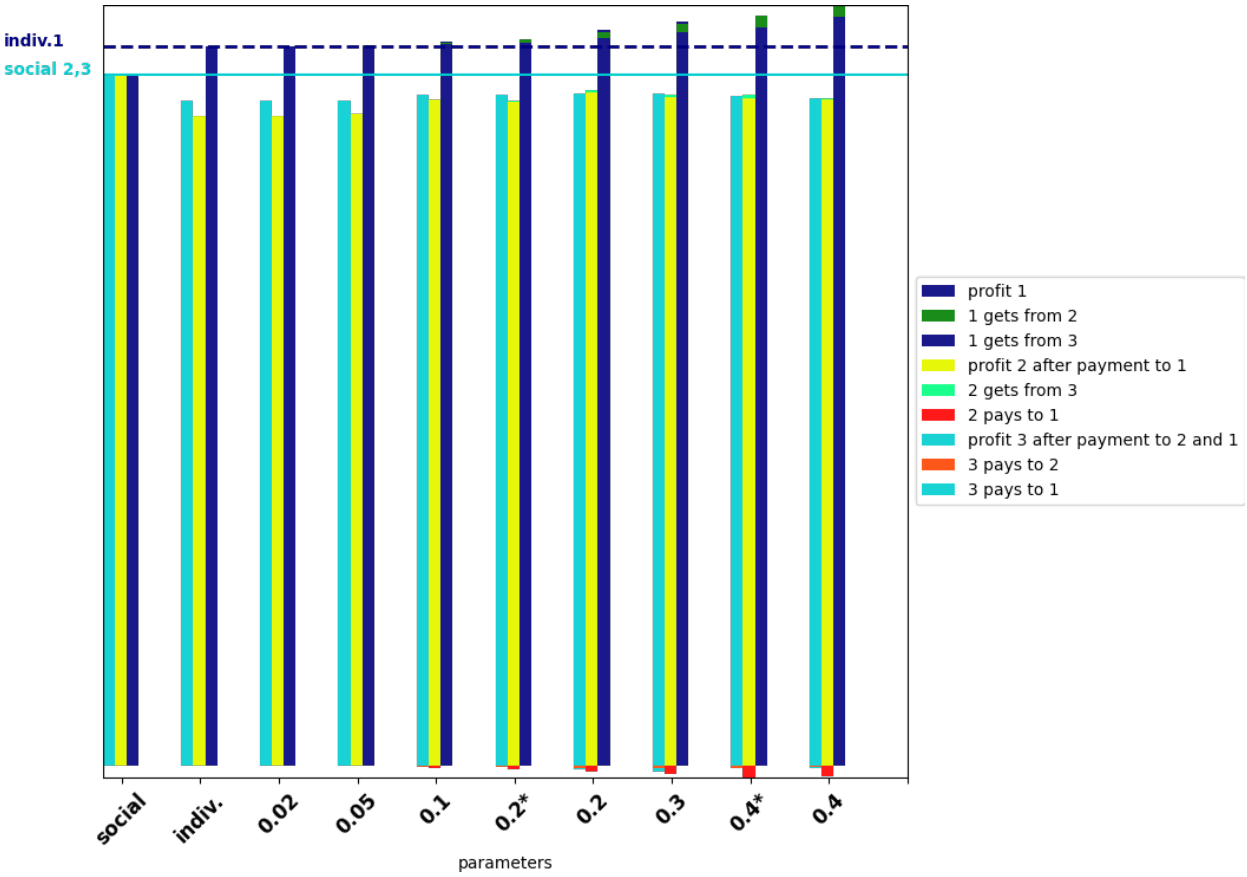

Figure 2: On the x-axis, each group with three columns represents the profit of levels 1, 2, and 3. The different group of columns correspond to the criterion employed in the optimization process: from left to right on the x-axis, these are the social optimum, the individualistic setting, and an increasing amount of profit sharing. We see that the profit-sharing mechanism is effective to recover more wealth from the cascade as a whole. With not too large transfers (the red bottom areas), when $\tau_{2\to1} = \tau_{3\to2} = 0.2$, level 1 makes more profit than in the individualistic setting, and levels 2 and 3 get the closest to the social policy. Larger transfers do not improve the situation for the downhill levels. When numbers are followed by a star, $\tau_{3\to1} = 0$, noting that tuning this parameter is more delicate, because there is an additional interplay within levels when $l = 3$ makes two transfers of profit.

the effective improvement of profit downstream. This re-scaling, transferring net margins with respect to the individualistic policy instead of gross values, has a significant impact in the numerical solution, at least with our data.

Figure 2 reports the differences in profit, compared to the social one. For each level and run, profits are presented as bars proportional to the profit obtained with the social policy (in the first group of bars), assimilated to 100%.

The individualistic policy, reported in the second group of bars, results in a increase for level 1, but decreases the profit of both levels 2 and 3. A policy will be acceptable for level 1 only if the gain is at least the individualistic profit. This corresponds to the dashed horizontal line in the graph. By contrast, levels 2 and 3 prefer policies that drive their profit as close as possible to the social one, whose level is indicated by the solid horizontal line in the graph. The remaining groups of bars correspond to different configurations of the profit sharing mechanism. Numbers in the abscissa indicate the value that was taken for $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2}$, except when the number is followed by a star, in which case $\tau_{3\to1} = 0$, and $\tau_{2\to1} = \tau_{3\to2} =$ the displayed number. Red bars in the bottom represent the payments done from the downhill levels uphill (always as a percentage of the social profit). When there is more than one color in the top of a bar, the area illustrates a transfer of profit from downhill levels. This is perceptible for example in the column labeled 0.2, where the profit of level 2 is increased by a transfer from level 3, and level 1 profit gets transfers from both levels 2 and 3. For each level $l \in \{1, 2, 3\}$, the bars with profit represent the gain, net from payments uphill. Therefore, when stacking on top the transfer from levels below (in a different color in the figure), the top of the bar corresponds to the final profit of the considered policy. Numeric values for the transfers can be found in the Appendix A, Table 3.

Figure 3 reports the water management for level l in the lth row of plots. The left, middle, and right columns correspond, respectively, to the social, individualistic and profit-sharing policies, with $\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.2$, the best parameters in Figure 2. We do observe that the agents at levels 1 and 2 withhold less water during periods of low prices, due to the profit-sharing mechanism. Moreover, we observe that the increase in profits for levels 2 and 3 come from not needing to release as much water without producing energy, relative to the individualistic solution. In other words, we observe some colaboration between the levels of the cascade.

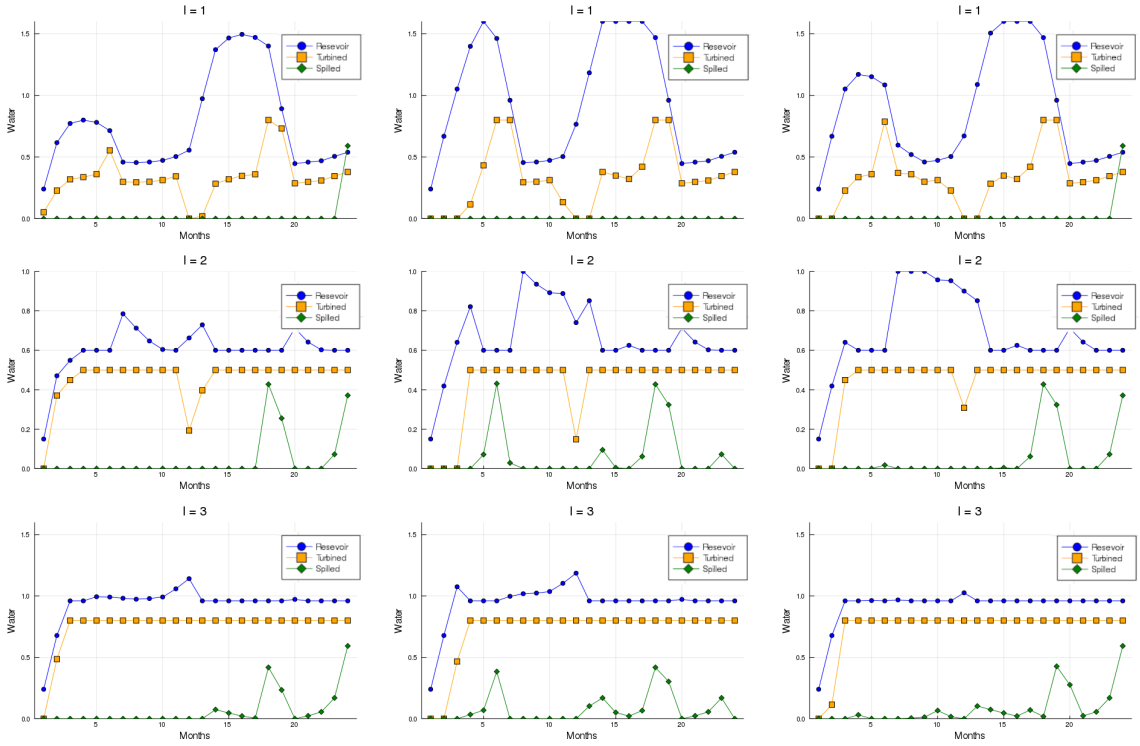

Figure 3: Water management of the cascade with social, individualistic and profit-sharing mechanisms (left, middle, right columns). With the individualistic approach (4), level 1 at the top withholds water until time $t = 5$, when the plant starts turbining to get high prices. There is more spillage with the individualistic approach, when compared to the other policies.

3 Nested stochastic optimization: individualistic approach

The trilevel formulations so far are deterministic. Problem (13) gives an idea on the changes induced by uncertainty. To start with, prices $\mathbf{\Pi}_s^t$ in the objective function are uncertain and given by S equiprobable scenarios, so we now have to deal with costs of the form $f_{l,s}^t$, for $s \in \{1, \ldots, S\}$. Regarding the stochastic analogue of the equality $A_l x_l = a_l - B_{l-1} x_{l-1}$, it involves the water balance constraint

$$v_l^{t+1} = v_l^t + \eta(-u_l^t - w_l^t + \boldsymbol{A}_l^t + \boldsymbol{u}_{l-1}^t + \boldsymbol{w}_{l-1}^t),$$

for inflows $\boldsymbol{A}_l^t$ that were considered deterministic so far. The constraint assumes a water travel time equal to one time period: the water released $(\boldsymbol{u}_{l-1}^t + \boldsymbol{w}_{l-1}^t)$ arrives to the reservoir downhill at time $t + 1$.

In a stochastic model, random inflows are specified by scenarios such as $\boldsymbol{A}_{l,r}^t$ and $\boldsymbol{A}_{l,s}^t$, for $r, s \in \{1, \ldots, S\}$ and all variables become indexed by scenarios. Suppose for a moment that the water travel time is equal to 1, then the water released by level $l-1$ at time t reaches level l at time $t+1$. In particular, the rth inflow scenario $\boldsymbol{A}_{l-1,r}^t$ resulted in decisions $\boldsymbol{u}_{l-1,r}^t$ and $\boldsymbol{w}_{l-1,r}^t$ and at level l this impacts the water

balance constraints. If the sth inflow scenario $\boldsymbol{A}_{l,s}^{t+1}$ occurs, the corresponding constraint will be

$$v_{l,s}^{t+1} = \boldsymbol{v}_{l,r}^{t} + \eta(-u_{l,s}^{t+1} - w_{l,s}^{t+1} + \boldsymbol{A}_{l,s}^{t+1} + \boldsymbol{u}_{l-1,r}^{t} + \boldsymbol{w}_{l-1,r}^{t})\,.$$

Note that scenarios r and s have no reason to be the same. In order to simplify the presentation, we consider below that the water travel time is zero (as opposed to the other modeling just explained), so that we deal with constraints of the form

$$v_{l,s}^{t+1} = \boldsymbol{v}_{l,s}^{t} + \eta(-u_{l,s}^{t+1} - w_{l,s}^{t+1} + \boldsymbol{A}_{l,s}^{t+1} + \boldsymbol{u}_{l-1,s}^{t+1} + \boldsymbol{w}_{l-1,s}^{t+1}) \tag{15}$$

with the same scenario for all levels (the technique can still be applied for positive travel times, but the notation becomes too cumbersome).

For clarity, note that we consider different scenarios r, s when the travel time of the water between reservoirs is positive. This is the reason we consider $\boldsymbol{u}_{l-1,r}^{t} + \boldsymbol{w}_{l-1,r}^{t}$ on the reservoir balance equation at stage $t+1$. In other words, the water released in stage t, just arrives downstream at stage $t+1$. When there is no travel time, the scenarios must be equal ($r = s$) and we consider the term $\boldsymbol{u}_{l-1,s}^{t+1} + \boldsymbol{w}_{l-1,s}^{t+1}$ on the balance equation at stage $t+1$. This is very intuitive, since the scenario is a state of the world on a given stage. Therefore, at the same stage, we have the same scenario. Analogously, for different stages, we could have different scenarios (states of the world).

Recall from (14) that individualistic profits are needed to chose ϕ therein and determine the actual transfer between levels. When dealing with scenarios the level interaction is quite intricate; in this section we explain the methodology for the uncertain version of (4), that is, when there is no transfer of profit between levels. The stochastic setting with profit being shared between consecutive levels is left for Section 4.

3.1 Computing individualistic two-stage policies

Since the first time step is considered deterministic in our model, both for the two-stage and the multi-stage cases, the specification of (15) for different time steps is

$$\begin{array}{rcll} v_{l}^{1} & = & \boldsymbol{v}_{l}^{0} + \eta(-u_{l}^{1} - w_{l}^{1} + \boldsymbol{A}_{l}^{1} + \boldsymbol{u}_{l-1}^{1} + \boldsymbol{w}_{l-1}^{1}) & \\ v_{l,s}^{t+1} & = & \boldsymbol{v}_{l,s}^{t} + \eta(-u_{l,s}^{t+1} - w_{l,s}^{t+1} + \boldsymbol{A}_{l,s}^{t+1} + \boldsymbol{u}_{l-1,s}^{t+1} + \boldsymbol{w}_{l-1,s}^{t+1}) & t = 1,\ldots,\mathtt{T}-1\,,\, s = 1,\ldots,S\,. \end{array} \tag{16}$$

With the two-stage paradigm, data is considered deterministic until a time when uncertainty reveals, all at once, until the end of the horizon. For presentation purposes, it is convenient to assume that uncertainty is fully revealed after the first time step, and that the whole path of inflows $\boldsymbol{A}_{l,s}^{t+1}$ becomes known at once, for $t = 1,\ldots,\mathtt{T}-1$ and each $s = 1,\ldots,S$. Then in (16) we deal with scenarios for $t \geq 2$. Adopting a notation that can be extended for more than two stages, we let

$$\begin{array}{rcll} x_{l}^{1} & := & \left(u_{l}^{t}, v_{l}^{t}, w_{l}^{t}, \text{slacks}_{l}^{t}\right) & \text{for } t = 1, \\ x_{l,s}^{2} & := & \left(u_{l,s}^{t}, v_{l,s}^{t}, w_{l,s}^{t}, \text{slacks}_{l,s}^{t}\right)_{t=2}^{\mathtt{T}} & \text{for each scenario } s = 1,\ldots,\,, \end{array}$$

respectively denote the first and second-stage variables. With this notation, choosing appropriate vectors and matrices, the relations in (16) can be represented in an abstract manner as

$$\begin{array}{rcll} A_{l}x_{l}^{1} & = & a_{l} - B_{l}x_{l-1}^{1} & \text{for constraints involving } t = 1, \\ A_{l,s}x_{l,s}^{2} & = & a_{l,s} - T_{l,s}x_{l}^{1} - B_{l,s}x_{l-1,s}^{2} & \text{for constraints involving } t = 2,\ldots,\mathtt{T}\,,\, s = 1,\ldots,S\,. \end{array} \tag{17}$$

For instance, taking $a_l = \boldsymbol{v}_l^0 + \eta \boldsymbol{A}_l^1$ and $A_l = B_l = [\eta \quad 0 \quad \eta \quad 0]$ gives the first equality in (16).

The starting point is once more the optimization problem at level l, written without any influence of the other levels. This corresponds to taking null matrices $B_{l,s}$ above and, hence, we look for solutions to

$$\left\{\begin{array}{ll} \min\limits_{x_l^1 \geq 0} & f_l^\top x_l^1 + \frac{1}{S}\sum\limits_{s=1}^{S} Q_{l,s}(x_l^1) \\ \text{s.t.} & A_l x_l^1 = a_l \end{array}\right. \quad \text{for recourse functions } Q_{l,s}(x_l^1) := \left\{\begin{array}{ll} \min\limits_{x_{l,s}^2 \geq 0} & f_{l,s}^\top x_{l,s}^2 \\ \text{s.t.} & A_{l,s}x_{l,s}^2 = a_{l,s} - T_{l,s}x_l^1\,. \end{array}\right.$$

When levels are organized in a cascade, since the RHS terms are modified by decisions taken in level $l-1$, recourse functions depend not only on the first-stage variable, but also on the decision taken at level $l-1$ for the same scenario realization. As a result, instead of $Q_{l,s}(x_l^1)$ as above, we now have the function $Q_{l,s}(x_l^1, x_{l-1,s}^2)$, that remains polyhedral and convex.

In a two-stage formulation, the trilevel individualistic problem to be solved is (the symbol "I2" refers to the individualistic model with two stages)

$$\begin{array}{lll} \text{find} \quad (x_1^{1^{\mathtt{I2}}}, x_2^{1^{\mathtt{I2}}}, x_3^{1^{\mathtt{I2}}}) \quad \text{such that} \quad x_l^{1^{\mathtt{I2}}} \quad \text{solves} & & \begin{cases} \min\limits_{x_l^1 \geq 0} & f_l^\top x_l^1 + \frac{1}{S}\sum\limits_{s=1}^{S} Q_{l,s}(x_l^1, x_{l-1,s}^{2^{\mathtt{I2}}}) \\ \text{s.t.} & A_l x_l^1 = a_l - B_l x_{l-1}^{1^{\mathtt{I2}}} \end{cases} \\ \text{where} \quad x_{l,s}^{2^{\mathtt{I2}}} \quad \text{solves} & Q_{l,s}(x_l^1, x_{l-1,s}^2) \quad := & \begin{cases} \min\limits_{x_{l,s}^2 \geq 0} & f_{l,s}^\top x_{l,s}^2 \\ \text{s.t.} & A_{l,s} x_{l,s}^2 = a_{l,s} - T_{l,s} x_l^1 - B_{l,s} x_{l-1,s}^2 \end{cases} \end{array}$$

for $l = 1, 2, 3$. (18)

In these problems, all terms involving a subindex $l-1 < 1$ are void.

Algorithm 2 puts in place a decomposition approach that we refer to as a "cascaded" L-shaped method, as it extends to the trilevel setting the well-known algorithm of [VW69]. We denote by $Q_{l,s}^k$ the current cutting-plane approximations for the recourse of each scenario and level, with the aggregated valued defined as

$$\mathfrak{Q}_l^k(x_l^1) := \frac{1}{S}\sum_{s=1}^{S} Q_{l,s}^k(x_l^1, x_{l-1,s}^{2k}) \quad \text{at iteration } k. \tag{19}$$

In (19), the left-hand side the dependency on $x_{l-1,1}^{2k}, \ldots, x_{l-1,S}^{2k}$ is dropped, for convenience.

In Algorithm 2, the loop parses $l \in \{1, 2, 3\}$, sequentially in the levels, and parallel with respect to scenarios (the second-stage subproblems can be solved independently). Replacing throughout $x_{l-1,s}^{2k}$ by the value computed at the previous iteration, $x_{l-1,s}^{2\,k-1}$ would yield a variant that is parallelizable also in the levels. Regarding convergence properties, by construction, the approximate recourse functions satisfy $Q_{l,s}^k(x_l^1, x_{l-1,s}^2) \leq Q_{l,s}(x_l^1, x_{l-1,s}^2)$ for all scenarios s, levels l and iterations k. Except for larger dimensionality, we are in a situation equivalent to the one for v_2 in item (ii) of Theorem 2.1. As a result, under the same assumptions on the linear programming solver, if $\varepsilon = 0$, after a finite number of iterations Algorithm 2 finds a solution to (18), the two-stage stochastic formulation of the individualistic trilevel problem.

3.2 Computing individualistic multi-stage policies

In the multi-stage paradigm, uncertainty is revealed gradually; the tth realization becomes known at the tth time stage. Accordingly, we shall deal with variables of the form

$$x_{l,s}^t \quad := \quad \left(u_{l,s}^t, v_{l,s}^t, w_{l,s}^t, \text{slacks}_l^t\right) \quad \text{for } t = 1, \ldots, \mathtt{T} \text{ and for each scenario } s = 1, \ldots, S,$$

and constraints will be like the right-most equality in (17), noting that all scenarios have the same realization at the first time stage, so $x_{l,s}^1 = x_l^1$ is deterministic.

Parsing all the branches of the scenario tree in a multi-stage setting, as it was done in (20) in Algorithm 2, is clearly impractical. Sampling algorithms like SDDP [PP91] are the methods of choice in multistage programming. However, for a cascade of nested optimization problems, with decisions from level $l-1$ impacting the RHS of the problem at level l, a straightforward application of the SDDP approach is also impractical. To be more precise, recall that each basic SDDP iteration consists in certain forward and backward passes (the details are given below). Suppose we start at level 1 with a standard SDDP iteration. Once the forward-backward iterates at level 1 are available, they become RHS scenario information for level 2, therefore modifying the scenarios seen by $l = 2$. A brute-force approach in this setting would run a standard SDDP method for $l = 1$ until it converges, and only afterwards move to level 2, running a separate SDDP for each RHS defined by the forward-backward iterates at level 1 and averaging the results. After satisfying a convergence criterion for each one of those runs at level 2, the brute-force

Algorithm 2 CASCADED L-SHAPED METHOD FOR TWO-STAGE STOCHASTIC TRILEVEL PROBLEMS. INDIVIDUALISTIC CASE

Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$ and $Q_{l,s}^k(\cdot) \equiv -M$ for $l = 1, 2, 3$ and $s = 1, \ldots, S$. For $l = 1$, let $x_{l-1}^{1k} = x_0^{11}$ and $x_{l-1,s}^{2k} = x_{0s}^{21}$ for $s = 1, \ldots, S$ be void elements.

REPEAT for $l = 1, 2, 3$:

First-stage master problem at level l: Given $y_{l-1,s}^k$ for all $s = 1, \ldots, S$ and $\mathfrak{Q}_l^k$ from (19),

$$x_l^{1k} \text{ solves } \begin{cases} \min\limits_{y \geq 0} & f_l^\top y + \mathfrak{Q}_l^k(y) \\ \text{s.t.} & A_l y = a_l - B_l x_{l-1}^{1k} . \end{cases}$$

Second-stage subproblems at level l: for each scenario $s = 1, \ldots, S$,

$$x_{l,s}^{2k} \text{ solves } Q_{l,s}(x_l^{1k}, x_{l-1,s}^{2k}) := \begin{cases} \min\limits_{y \geq 0} & f_{l,s}^\top y \\ \text{s.t.} & A_{l,s} y = a_{l,s} - T_{l,s} x_l^{1k} - B_{l,s} x_{l-1,s}^{2k} . \end{cases} \tag{20}$$

Let $\lambda_{l,s}^k$ denote the optimal multiplier vector associated with the equality constraints.

Model improvement for level l: For $s = 1, \ldots, S$ the linearization

$$\ell_{l,s}^k(x_l^1, x_{l-1,s}^2) := Q_{l,s}(x_l^{1k}, x_{l-1,s}^{2k}) + (\lambda_{l,s}^k)^\top T_{l,s}(x_l^1 - x_l^{1k}) + (\lambda_{l,s}^k)^\top B_{l,s}(x_{l-1,s}^2 - x_{l-1,s}^{2k}) \tag{21}$$

improves the approximate recourse functions and its expected value,

$$Q_{l,s}^{k+1}(x_l^1, x_{l-1,s}^2) := \max\left\{ \ell_{l,s}^k(x_l^1, x_{l-1,s}^2), Q_{l,s}^k(x_l^1, x_{l-1,s}^2) \right\} , \quad \text{and } \mathfrak{Q}_l^{k+1}(x_l^1) \text{ defined like in (19).}$$

STOPPING TEST. Stop if for all the levels $\dfrac{1}{S} \displaystyle\sum_{s=1}^{S} Q_{l,s}(x_l^{1k}, x_{l-1,s}^{2k}) - \mathfrak{Q}_l^{k+1}(x_l^{1k}) \leq \varepsilon$.

Otherwise, set $k = k + 1$ and go to **REPEAT**. □

method would move to level 3. Since such approach is too cumbersome, in this section we show how to perform calculations in a manner that is computationally efficient.

Instead of sequentially applying SDDP for level l and, when a convergence criterion is reached, moving to $l+1$, we perform one SDDP forward and backward passes for all the levels, and then iterate across levels. Recall that decisions taken upstream modify the downhill ones only via the RHS equality constraints, as in (17). In Algorithm 3 the key is in suitably transferring to level l information obtained with the SDDP iteration done at level $l-1$. This is done by transporting cuts along levels.

The multi-stage formulation inevitably requires rather involved notation, that we gradually introduce for clarity. There are time stages $t \in \{1, \ldots, \mathtt{T}\}$ and S stage-wise independent scenarios referred to by a subindex s or $r \in \{1, \ldots, S\}$, having $\mathtt{T}$ components. Below, the notation with subindex s refers to what is called a forward SDDP scenario, while r refers to scenarios in the backward SDDP pass. Note that we can replace the stage-wise independence with a markovian structure on the formulations, in which case, all calculations can be performed with minor changes.

With respect to the notation in the previous section, given a scenario s,

$$\text{at stage } t \left\{ \begin{array}{lllll} \text{the former here-and-now variable} & x_l^1 & \text{corresponds to} & x_l^{t-1} \\ \text{the former recourse variables} & x_{l,s}^2 & \text{correspond to} & x_{l,s}^t \end{array} \right. \text{ in the multi-stage setting.}$$

Consider first the simple setting of disconnected levels and the following shorter notation for the feasible sets

$$F_{l,s}^t(x_l^{t-1}) := \left\{ y \geq 0 : \;\; A_{l,s}^t y = a_{l,s}^t - T_{l,s}^t x_l^{t-1} \;\; \right\},$$

for $l = 1, 2, 3$, $t = 1, \ldots, \mathtt{T}$ and $s = 1, \ldots, S$, and where x_l^0 is a given data, sometimes called the tendency. Note that the input of $F_{l,s}^t(\cdot)$ does not depend on the scenario, since it can be evaluated for any iterate from state $t-1$ from any scenario, but the set-valued function $F_{l,s}^t$ does depend on s. Letting $\mathbb{E}$ stand for the expectation operator, a nested representation for this multi-stage problem is

$$\min_{x_l^1 \in F_l^1(x_l^0)} {f_l^1}^\top x_l^1 + \mathbb{E}\left[\min_{x_l^2 \in F_l^2(x_l^1)} {f_l^2}^\top x_l^2 + \mathbb{E}\left[\ldots + \mathbb{E}\left[\min_{x_l^{\mathtt{T}} \in F_l^{\mathtt{T}}(x_l^{\mathtt{T}-1})} {f_l^{\mathtt{T}}}^\top x_l^{\mathtt{T}} \right] \ldots \right] \right]. \tag{22}$$

In a Dynamic Programming formulation, each bracket above represents the recourse, often called cost-to-go, or future cost function, shortened to FCF from now on. In particular, the FCF at time t represents the costs of all the decisions taken between $t+1$ and $\mathtt{T}$.

The basis of the SDDP approach is to define approximations at iteration k moving first forward in (22), from t to $t+1$, to find feasible points $\hat{x}_{l,s}^{tk}$ for the sampled scenario $s = s^k = (s_1^k, \ldots, s_{\mathtt{T}}^k)$. Then (22) is parsed from right to left, moving backwards from $t+1$ to t, generating linearizations along all the branches of the given scenario, and the process is repeated with k replaced by $k+1$. The points $\hat{x}_{l,s}^{tk}$ computed in the forward pass are available for all $t = 1, \ldots, \mathtt{T}$. At $t = \mathtt{T}$, given the forward vector $\hat{x}_{l,s}^{\mathtt{T}-1,k}$, the backward pass computes

$$x_{l,r}^{\mathtt{T}k} \text{ solving } Q_{l,r}^{\mathtt{T}}(\hat{x}_{l,s}^{\mathtt{T}-1,k}) := \left\{ \begin{array}{ll} \min & (f_{l,r}^{\mathtt{T}})^\top y \\ \text{s.t.} & y \in F_{l,r}^{\mathtt{T}}(\hat{x}_{l,s}^{\mathtt{T}-1,k}), \end{array} \right.$$

for all $r = 1, \ldots, S$. Similarly to (21), the solution process provides a linearization that improves the current piecewise affine function $Q_{l,s}^{\mathtt{T}k}$. An average of those cutting-plane models gives the expected value $\mathfrak{Q}_l^{\mathtt{T},k}$, the FCF for the backward problem at $t = \mathtt{T}-1$. Proceeding further for any stage $t = \mathtt{T}-1, \mathtt{T}-2, \ldots$, for all $r = 1, \ldots, S$ the backward iterate

$$x_{l,r}^{tk} \text{ solves } Q_{l,r}^{tk}(\hat{x}_{l,s}^{t-1,k}) := \left\{ \begin{array}{ll} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,s}^{t-1,k}). \end{array} \right.$$

Letting $Q_{l,r}^{\mathtt{T}+1,k} \equiv 0$, the formulation above is also valid for $t = \mathtt{T}$. The backward pass generates linearizations that improve the cutting-plane models defining the FCF to be used in the next forward pass, $\mathfrak{Q}_l^{t+1,k+1}$. Convergence of such procedure to a solution of the multi-stage problem, with probability one, can be found in [Sha11], as well as the required assumptions for the result to hold (on the sampling and on conditions on the linear programming solver similar to those in Theorem 2.1).

When feasible sets are connected stagewise but not between levels, the FCF depends only on the decision taken for the considered scenario and stage, at the current level l. When levels are connected in a cascade, the recourse functions of level l depend on decisions taken at level $l-1$. As explained for the two-stage case, using the multi-stage notation and for a scenario $r = 1, \dots, S$ and $l = 1, 2, 3$,

$$\left(x_l^2, x_{l-1,r}^2\right) \text{ from Algorithm 2 becomes } \left(x_l^{t-1}, x_{l-1,r}^t\right),$$

and, similarly to (17), the RHS dependencies change the feasible sets to

$$F_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t) := \left\{ y \geq 0 : \;\; A_{l,r}^t y = a_{l,r}^t - T_{l,r}^t x_l^{t-1} - B_{l,r}^t x_{l-1,r}^t \;\; \right\}.$$

Hence, with connected levels, $Q_{l,r}^{tk}$ is a function of $\left(x_l^{t-1,k}, x_{l-1,r}^{tk}\right)$. Note that that we use $x_l^{t-1,k}$ instead of $x_{l,r}^{t-1,k}$, because the value of $x_l^{t-1,k}$ can come from any scenario at stage $t-1$. The two-stage case is similar to the last bracket in (22). Carrying on the parametric dependencies backwards by reasoning recursively based on the two-stage case, we see that the recourse function at stage t and level l depends

$$\begin{array}{lll} \text{on} & (x_l^{t-1}, x_{l-1,r}^t), & \text{through the feasible set } F_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t), \\ \text{on} & Z_{l-1}^{t+1} := \left(x_{l-1,s}^{t'}, s = 1, \dots, S, t' = t+1, \dots, \mathtt{T}\right), & \text{through the future cost function}. \end{array} \tag{23}$$

With respect to the two-stage setting, the main difference is in the dimensionality increase of the arguments of the recourse function. As such, the trilevel individualistic problem is an extension of (18) to the multi-stage setting, with the solutions $x_{l,s}^{t^{\text{IT}}}$ solving nested problems of the form

$$Q_{l,r}^t(x_{l,s}^{t-1^{\text{IT}}}, Z_{l-1}^{t^{\text{IT}}}) = \begin{cases} \min & {f_{l,r}^t}^\top y + \frac{1}{S}\sum_{s=1}^S Q_{l,s}^{t+1}(y, Z_{l-1}^{t+1,^{\text{IT}}}) \\ \text{s.t.} & y \in F_{l,r}^t(x_{l,s}^{t-1^{\text{IT}}}, x_{l-1,r}^{t^{\text{IT}}}), \end{cases}$$

for each scenario $r, s = 1, \dots, S$, stage $t = 1, \dots, \mathtt{T}$ and level $l = 1, 2, 3$.

At iteration k of the cascaded SDDP method, by definition of Z in (23), the identity

$$Z_{l-1}^{t,k} = (x_{l-1,1}^{tk}, \dots, x_{l-1,S}^{tk}, Z_{l-1}^{t+1,k})$$

holds. So at stage t the recourse function is computed at

$$(x_l^{t-1,k}, x_{l-1,1}^{tk}, \dots, x_{l-1,S}^{tk}, Z_{l-1}^{t+1,k}) = (x_l^{t-1,k}, Z_{l-1}^{t,k}),$$

and backward problems yield

$$x_{l,r}^{tk} \text{ solving } Q_{l,r}^{tk}(\hat{x}_{l,s}^{t-1,k}, Z_{l-1}^{t,k}) := \left\{ \begin{array}{ll} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,s}^{t-1,k}, x_{l-1,r}^{tk}), \end{array} \right.$$

where $\mathfrak{Q}_l^{t+1,k}$ is the expected future cost function.

The solution algorithm is given in Algorithm 3.

It is worth noting that the forward samples are common for all levels, which is quite natural if one has the brute-force solution procedure in mind. Also, the cut calculation performed at the backward pass is entirely based on the two-stage case. Because of the dependence on the iterates computed at level $l-1$, cuts have (much) larger dimension in the cascaded setting.

Since at the top level the past vectors x_{l-1}^{tk} are void, when $l = 1$ Algorithm 3 boils down to a standard SDDP iteration. Under the same assumptions as in [Sha11, Proposition 3.1], with probability one, the forward step at $l = 1$ produces an optimal policy after finitely many iterations. As a result, with probability one and for sufficiently large k, the cutting-plane models in our cascaded SDDP represent well the relevant parts of the future costs at level 1. This property does not suffice to ensure convergence in the trilevel setting, however. The reason bears some resemblance with risk-averse forms of SDDP, to solve problems as in (22), with the expectation operator replaced by a risk measure, see [KM14; Sha11; GR12]. As noted in [Sha11, Remark 5], sampling makes the upper bound v_1^k random. When passing to level $l = 2$, the lower

Algorithm 3 CASCADED SDDP METHOD FOR MULTI-STAGE STOCHASTIC TRILEVEL PROBLEMS. INDIVIDUALISTIC CASE

Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$. For $l = 1, 2, 3$, $s = 1, \ldots, S$ and $t \geq 2$, let $Q_{l,s}^{t,k} \equiv -M$. For $t = 1$, all levels and scenarios, the initial tendency $\hat{x}_{l,s}^{t-1,k}$ is given. For $l = 1$, the vectors $x_{l-1,s}^{tk}$ are void.

Sampling. Sample a scenario $s_t^k \in \{1, \ldots, S\}$ for each $t = 1, \ldots, \mathtt{T}$. To simplify notation recall that all scenarios at $t = 1$ are assumed to be the same.

REPEAT For $l = 1, 2, 3$:

Forward pass. For each $t = 1, \ldots, \mathtt{T}$ and $s = (s_1^k, \ldots, s_{\mathtt{T}}^k)$, compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,s}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,s}^{t-1,k}, x_{l-1,s}^{tk}) \,. \end{cases}$$

Init. Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

Iterate Across Stages. For $t = \mathtt{T}, \ldots, 2$.

Cut computation. For $r = 1, \ldots, S$, solve

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,s}^{t-1,k}, x_{l-1,r}^{tk}) \,. \end{cases}$$

Obtain subgradients such that for all $x_l^{t-1}, x_{l-1,r}^t$ and Z_{l-1}^{t+1} the value function $Q_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t, Z_{l-1}^{t+1})$ lies above

$$\begin{array}{lll} Q_{l,r}^t(x_{l,s}^{t-1,k}, x_{l-1,r}^{t,k}, Z_{l-1}^{t+1,k}) & + & (\lambda_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l,s}^{t-1,k}) \qquad + \\ (\mu_{l-1,r}^{t,k})^\top (x_{l-1,r}^t - x_{l-1,r}^{t,k}) & + & (\nu_{l-1}^{t+1,k})^\top (Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}). \end{array}$$

Cut Aggregation. Average the cuts in (3) to obtain a cut such that $\mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, Z_{l-1}^t)$ lies above

$$\mathfrak{Q}_{l,r}^t(x_{l,\omega}^{t-1,k}, Z_{l-1}^{tk}) + (\phi_l^{t-1,k})^\top (x_l^{t-1} - x_{l,\omega}^{t-1,k}) + (\rho_{l-1}^{tk})^\top (Z_{l-1}^t - Z_{l-1}^{tk}). \tag{24}$$

Define $\mathfrak{Q}_{l,r}^{t,k+1}$ as a maximum between $\mathfrak{Q}_{l,r}^{t,k}$ and (24).

Upper bound. Set $v_l^k = \sum_{t=1}^{\mathtt{T}} {f_{l,s}^t}^\top \hat{x}_{l,s}^{tk}$ for the estimation of the upper bounds via average, where $s = (s_1^k, \ldots, s_{\mathtt{T}}^k)$.

Lower bound. Set u_l^k as the optimal value of the subproblem at $t = 1$ solved on the backward pass for estimation of the lower bounds via average.

STOPPING TEST. Stop if the average lower bound and average upper bound for all levels are close enough or the lower bounds stabilized. Otherwise, set $k = k + 1$ and go to the **Sampling** step again. □

bound u_2^k is also random, because it depends on cuts involving forward-backward iterates from level 1. The lower bound at level 3 is also random and, therefore, no convergence result is available in this setting. However, in practice we observe small gaps, after averaging.

Algorithm 3 is essentially an efficient implementation of the brute-force algorithm described at the beginning of this section. The brute-force approach is not computationally practical because the formulation at level 2 depends on the forward path at level 1 and the decisions taken at level 1 associated with the forward path. Therefore, there are at least as many instantiations of problems at level $l = 2$ as there are forward paths. In other words, there are exponentially many problems at level 2. Instead of solving separately each problem at level 2 for each forward path at level 1 we employ the floating cuts such that, when a new sequence of forward-backward iterates is obtained at level 1, the cuts at level 2 already provide a valid lower bound for the SDDP problem at level 2 associated with the new forward path from level 1. Thanks to this feature, we do not need to solve the SDDP problem at level 2 from scratch. Analogously, the cuts at level 3 provide a valid lower bound given the new forward-backward iterates at level 2.

For a problem with four stages and three scenarios ($\mathtt{T} = 4$ and $S = 3$), the diagram in Figure 4 illustrates with solid lines how information flows from level l (up) to level $l + 1$ (bottom) Algorithm 3. For each level, dotted boxes indicate the path of scenarios sampled in the forward pass, given by $s = (0, 3, 2, 2)$ in the figure. Since the same path of scenarios is used for all levels, dotted boxes have the same position in the top and bottom rows. The dotted lines connecting boxes horizontally represent how information is transmitted in the same level, between time stages. Those lines have no arrows because information goes both ways, forward and backwards, as in the SDDP passes. Notice also that all boxes are connected from stage to stage. When moving forward in time, decisions from box $l, t = 2, s = 3$ go to all scenarios at $l, t = 3$, while when moving backwards, cuts from all boxes at $l, t = 3$ go back to $l, t = 2, s = 3$.

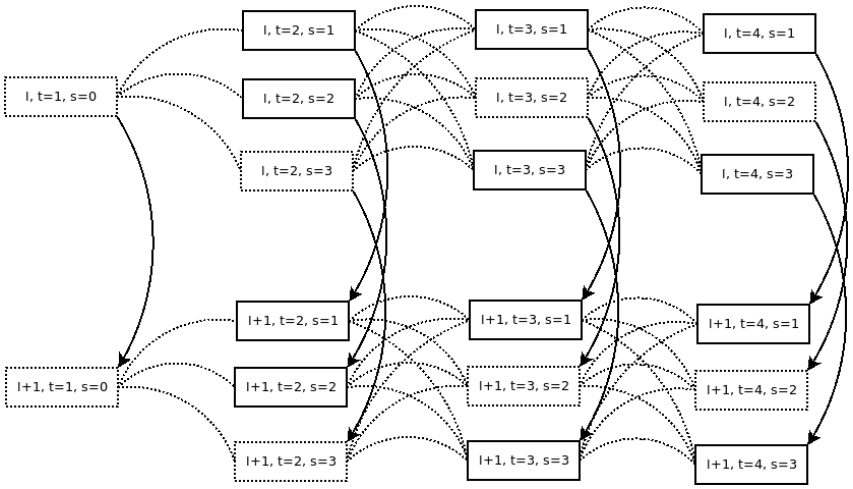

Figure 4: Illustration of the flow of information in Algorithm 3.

3.3 Numerical assessment

We designed a stochastic variant of our toy problem (13), with uncertainty in the prices $\mathbf{\Pi}^t$ and inflows $\boldsymbol{A}_l^t$. The respective considered scenarios are shown in Figure 5.

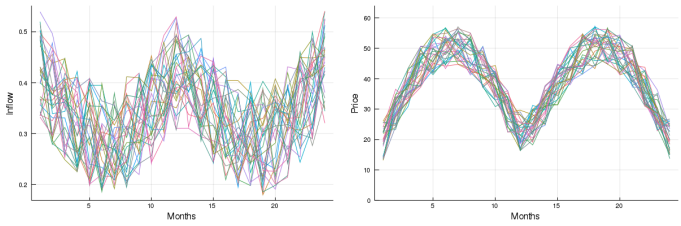

Figure 5: Scenarios for the water inflow and price. Inflows are shown as percentages of reservoir's volume. The inflow around December is a fraction of the maximum volume followed by a dry period around the middle of the year. Prices follow an inverse pattern. Recall that the deterministic values taken for price and inflow in (13) are stagewise averages of these scenarios.

The dimensions of the cascade are the same of the deterministic case, as well as the initial volumes. The individualistic policies obtained by the cascaded SDDP method presented in Algorithm 3 are compared to the social policies obtained with the traditional SDDP method for managing the cascade jointly. Professional SDDP software typically exploits parallelization. This is not the case with our implementation, which took up to 8 hours to produce the output reported below. We run 100 forward-backward iterations on each level and initialize the cuts at $l = 1$ to make the process faster. All the obtained gaps are smaller than 4%, relative to the lower bound.

We simulate the cascade operation with each policy, for new out-of-sample 100 scenarios, the corresponding mean profit for each level is reported in Table 2.

Policy Type	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$	Tot. Profit
Social (Standard SDDP)	336.89	439.16	710.05	1486.12
Individualistic (Cascaded SDDP)	+4.20%	-6.13%	-2.86%	-2.22%

Table 2: Comparison of expected profits for each level relative to the social policy. The qualitative behavior of the profits is like in the deterministic case: the hydroplant at $l = 1$ earns more and the hydroplants downhill earn less. Also, the total wealth obtained from the cascade decreases.

The mean reservoir operation and its standard deviation are reported for each level in Figure 6, respectively in solid and dashed lines. With the social policy, the spillage observed at the end of the simulation is just an evidence of the end-of-horizon effect, that does not affect the profits. With the individualistic policy, similarly to the deterministic model, the top hydroplant does not deplete its reservoir at initial times, saving water for it to be released when prices are high. The pattern of each policy is similar to one observed in the deterministic case reported in Figure 3.

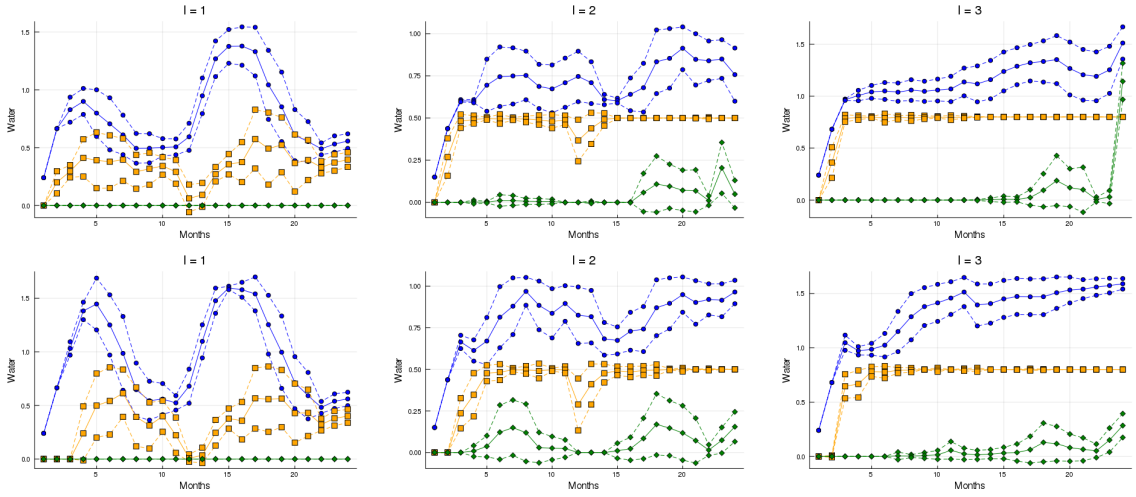

Figure 6: Water management of the cascade with social and individualistic policies (top and bottom rows) in the stochastic setting, computed with three different approaches. Circles represent volume, squares turbined water and "diamond" spillage. Given the lines of the same color, the dashed lines correspond to the larger and smaller variation in the solutions of the three methods.

4 Sharing mechanism between neighbors only

We now extend the multi-stage procedure to deal with the mechanism of profit sharing. With respect to the deterministic case, the setting is slightly less general, as here we assume the transfer is done only between consecutive levels (this amounts to taking $\tau_{3\to 1} = 0$ in the deterministic formulation). For the cascade in Figure 1, we now consider dependencies represented by both the left and right arrows, only that level 3 shares profit with level 2, but not with level 1 (in the figure, the left arrow with the label $\tau_{3\to 1}$ is not present).

The iterative procedure follows the rationale in the previous sections, defining cuts based on extended variables, as in (23), only that now we adopt a handy symbolic representation. The mechanism is described below in an informal style to avoid heavy and cumbersome notation. A more precise statement of the algorithm is given in Appendix B.

The setting considered in this section is more general than the one considered in Section 3, since taking all transfers to be zero in this section, we obtain the same problem as in Section 3. Since the brute-force approach was already infeasible for Section 3, it is not practical for the developments of this section as well. Note that the symbolic representation is just a strategy to present the algorithm and to facilitate its implementation.

4.1 The concept of floating cut

The multi-stage individualistic model gives a hint on the difficulties that need to be tackled when there is a hierarchy of three nested multi-stage programs with transfer of profit to the next level upstream. The challenge in the more general setting considered here starts with defining an extended variable, from which the parametric dependencies in the FCF can be written down, to define the linearizations, or "cuts". Along the lines in (23), we need to detect connections between the current decision variable (say x_l^{t-1}) and decision variables of other levels (say Z_{l-1}^t).

The difference here is that, instead of trying to figure out all the (nontrivial) dependencies that can happen by ourselves, we put in place a symbolic code that detects automatically those relations and defines a "floating cut". The procedure starts representing the forward and backward linear programs that are solved for each level in a manner similar to a modeling language. For a given optimization problem, the representation stores, in a human-readable format, three data structures with relevant information. A first structure deals with variables, distinguishing decisions from parameters (a certain flag is set to 0 or 1),

specifying attributes such as type (continuous, binary), bounds, name, and storing the actual value of the variable in question. A second data structure represents the linear expressions appearing in the optimization problem (a real number and a list of pairs of real numbers and instances of the data structure variable). The third structure contains a linear expression for the objective function and a list of pairs of linear expressions and integers to represent the constraints in the optimization problem under consideration.

With this symbolic representation at hand, and its distinction between parameters and decisions, it is possible to compute values for the latter, given the values of the former. The same solution process yields optimal dual variables for the constraints, and these values are stored in the first data structure, with the problem variables. A floating cut is the symbolic expression of a linearization like (10) in Algorithm 1. Namely, an affine relation given by the Benders cut that results from fixing the parameters at their current values in the data structure. Since the floating cut is a linear expression of all the right-hand side parameters involved in the optimization problem, it can be symbolically represented by means of the second data structure. At every iteration, knowing the parameters of the optimization problem, a specific instance is obtained, and its value is inserted as a new constraint in the third data structure of other optimization problems (those for which the current decision variable appears as a parameter).

The sophisticated construction of floating cuts is fundamental to manipulate efficiently the huge amount of optimization problems and linearizations involved in the multi-stage three-level setting. The naming is justified by the following observation. When a specific subproblem is to be solved, the portions of the linear expressions associated with parameters are reduced to a number, called the parametric value of the linear expression, based on the values of those parameters. When the values of the parameters change, the parametric value of the linear expressions change too. In turn, this change is interpreted as the free term of the linear constraints having "floated" to another level or stage.

As we explain now with an example, before floating a cut we might also need to update its value. Consider a SDDP problem with $\mathtt{T} = 3$ and $S = 2$ (only one level) and suppose iterations start from $\mathtt{T}$, backwards to $\mathtt{T} - 1$. The decision variables at $t = 3$ and $s = 1, 2$ are x_s^t and the parameter at $t = 3$ is the forward decision at $t = 2$, denoted by $\hat{x}^t$ (for this variable we drop the scenario subindices for convenience). The cuts computed on any scenario at $t = 3$ are functions of the parameter, the forward decision $\hat{x}^t$. By this token, for a subproblem at $t = 2$, the forward decision $\hat{x}^3$ will be considered as a parameter. But when $t = 2$, the parameter is x_1^2 if $s = 1$, and x_2^2 if $s = 2$. For this reason, the value of the floating cut computed at $t = 3$ needs to be updated, replacing the parameter $\hat{x}^3$ by the variable x_s^2 for each subproblem scenario. A similar translation step needs to be performed to account for the fact that the forward decision taken at $t = 1$ is a parameter for all subproblems at $t = 2$.

Suppose we represent symbolically the subproblems of a multi-stage stochastic problem and that we declare symbolically all the external parameters of each problem at each level, stage, and scenario. When making this symbolic declaration we do not know the dependencies of the FCF of that specific subproblem. However, we can always start with a large negative number as a valid approximation. Given the values of all the RHS parameters on the symbolic representation, a symbolic expression for the cut can be derived, which depends on all RHS parameters found on the symbolic representation. Whenever the values of the RHS parameters are updated, the symbolic linearization, that we named the "floating" cut, is updated too.

This symbolic representation of the cuts, instead of dealing with the usual matrix- and index-based representations, suffices to detect non trivial parametric dependencies for the multi-stage case, as long as calculations are carried out in the correct order. To understand this issue, let us explore the two-stage individualistic setting as an example. Assume we have forward-backward iterates at $l = 1$. The first cuts computed at $l = 2$ are those corresponding to the final time index. Each scenario subproblem at the final time $\mathtt{T}$ depends on x_l and $x_{l-1,s}^2$. The symbolic cuts computed at each scenario would be a linear function of both x_l and $x_{l-1,s}^2$. After averaging the symbolic expressions for all scenarios, we would recognize that the FCF at the first stage is a function of x_l, $x_{l-1,1}^2$, $\ldots$, $x_{l-1,S}^2$. The same reasoning applies to the multi-stage case, except that at stage $\mathtt{T} - 1$ we would have to search for new parameters on the symbolic representation of the subproblems at $\mathtt{T} - 1$ that come from subproblems at stage $\mathtt{T}$.

We emphasize that to compute the general affine expression of the floating cut, we need the values of all parameters and we need to possibly update the list of RHS parameters as dependencies appear along the process. In this sense, the individualistic cascaded SDDP Algorithm 3 first gets values of external parameters making a forward-backward pass at $l = 1$. When making the forward-backward pass at $l = 2$,

all the correct parametric dependencies show up naturally, which also gives the values of the external parameters at $l = 3$ and so on. Therefore, although hard to express precisely, the mechanism is not difficult to implement.

The procedure can fail if cuts are computed in the wrong order. More specifically, if $\mathtt{T} = 3$ and the first cut is computed at $t = 2$, calculations would be made without the parametric dependencies at the final time $t = 3$ and all the FCF estimations arriving at $t = 1$ would be wrong. For the multi-stage problems we solve, it is enough to just follow the standard iterations, but always start computing cuts at the last stage. This is not an issue for the traditional SDDP method: in the parametric symbolic view, it amounts to fixing the values of all the RHS parameters. The corresponding contribution of the fixed parameters on the floating cuts is zero and, hence, our approach is a generalization of the well-known SDDP algorithm.

These explanations should make it clear that rather than struggling to express the correct RHS parametric dependencies, the real issue is to organize the symbolic calculations in the correct order, starting from the final time T. In our trilevel problem, the order is clear since decisions are sequential in nature and, under reasonable assumptions, the initial lower bounds for value functions remain valid for bounded values of the external RHS parameters.

4.2 Computing policies with profit sharing

Having outlined the general procedure, we now focus on the sharing mechanism when dealing with nested SDDP problems. To fix ideas, suppose that SDDP problem 1 influences SDDP problem 2 and vice-versa (in the individualistic setting, SDDP problem 1 affects SDDP problem 2, which affects SDDP problem 3, but not the other way round). Dropping unnecessary indices, we let the corresponding value functions be defined as

$$v_1(x_2) = \min_{x_1} f_1(x_1, x_2) \text{ s.t. } x_1 \in X_1(x_2),$$

and

$$v_2(x_1) = \min_{x_2} f_2(x_2, x_1) \text{ s.t. } x_2 \in X_2(x_1)\,.$$

Typically, the solution to such a pair of problems is addressed by computing a generalized equilibrium, for example by iterating over the best-response of one player, given the other players' strategies are fixed. See [SMK18]. When the pair of problems at hand is simple, the best-response iteration is easy to implement and might converge to an equilibrium. When dealing with a pair of multi-stage stochastic problems, the situation is much less straightforward. Even the computation of the best response given the strategies of the other players is a hard task. Floating cuts are very useful in this setting, because linearizations computed for a given x_2^k can be carried over to another iterate x_2^{k+1}. Thanks to the floating cuts, a best-response iterative procedure is possible in the context of multi-stage stochastic equilibrium problems as we further explain now.

Similarly to (23), the tuple $Z_l = \{x_{l,s}^t : t = 1, \ldots, \mathtt{T}, \quad s = 1, \ldots, S\}$ represents feasible forward-backward iterates for each level l. Additionally, Z_{-l} is the vector of tuples referring to levels other than l; in particular, $Z_{-1} = (Z_2, Z_3)$. For the trilevel problem feasible tuples are obtained by sequentially making a forward-backward pass at $l = 1, 2, 3$, in order. The tuple Z_l is random because it depends on the scenarios sampled to perform the forward passes.

Recall that the parameter $\tau_{\to l}$ is the fraction of cost being transferred to level $l \in \{1, 2\}$ from level $l + 1 \in \{2, 3\}$. Accordingly, the symbolic declaration of the rth scenario subproblem at stage t and level $l = 1, 2$ is given by the expression below:

$$Q_{l,r}^t(x_l^{t-1}, Z_{-l}) := \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1}(y, Z_{-l}) + \tau_{\to l}\mathfrak{U}_{l,r}^t(y, Z_{-l}) \\ \text{s.t.} & y \in F_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t). \end{cases}$$

Notice that we now have two functions in the objective function. The first one, $\mathfrak{Q}$, deals with the usual nested Dynamic Programming scheme similar to (22). The second function, $\mathfrak{U}$, is specific to the sharing mechanism in our proposal. A closer inspection of the symbolic representation above reveals some additional features. To begin with, the first argument in the recourse function, x_l^{t-1}, has no scenario subindex because its value can come from any scenario at stage $t-1$ (as in the dotted lines in Figure 4). By contrast, the RHS value $x_{l-1,r}^t$, defining the feasible set, comes necessarily from the rth scenario at level

$l-1$ (as in the full lines in Figure 4). In addition, the cut computed from this representation is an affine function of Z_{-l} as well as of x_l^{t-1} and $x_{l-1,r}^t$.

Along iterations, the only information that changes in the symbolic representation is the piecewise affine approximations $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$ and $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$, which initially are set to a fixed negative number, sufficiently large. Being the well-known FCF within the SDDP problem at level l, $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$ is the usual function updated in a forward-backward pass. The new function $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$, represents the instantaneous cost realized at stage t and scenario r from level $l+1$. As such, for $l=2$, the new function is a piecewise affine approximation of

$$U_{l+1,r}^t(x_{l+1}^{t-1}, x_{l,r}^t) := \left\{ \begin{array}{ll} \min & {f_{l+1,r}^t}^\top y \\ \text{s.t.} & y \in F_{l+1,r}^t(x_{l+1}^{t-1}, x_{l,r}^t)\,, \end{array} \right.$$

and, for $l=1$, it is an approximation of

$$U_{l+1,r}^t(x_{l+1}^{t-1}, x_{l+2}^{t-1}, x_{l,r}^t) := \left\{ \begin{array}{ll} \min & {f_{l+1,r}^t}^\top y + \tau_{\to(l+1)} U_{l+2,r}^t(x_{l+2}^{t-1}, x_{l,r}^t) \\ \text{s.t.} & y \in F_{l+1,r}^t(x_{l+1}^{t-1}, x_{l,r}^t)\,. \end{array} \right.$$

Notice that the notation $\tau_{\to(l+1)}$ is different from the one for the deterministic setting, where we would have the indices $3 \to 1$, for instance. The reason is that, since we just have profit sharing between immediate levels, it is clear that $\tau_{\to(l+1)}$ refers to a transfer from $l+2$ to $l+1$.

We are now dealing with two different polyhedral approximations, whose update needs to be done in a manner slightly different from the usual SDDP. The following list explains the procedure step by step. Algorithms 4 to 9, given in the Appendix B, respectively correspond to pseudo-code for items 1-7 below. Note that the Algorithms 4 to 9 are split and numbered such that they fit into the pages nicely, and not such that they match the numbering of the items below.

1. First, a sequence of forward samples for each level is generated and the tuples (Z_1, Z_2, Z_3) are obtained by making a forward-backward pass at each level *without adding any cuts*. At this initialization phase, this is crucial to detect the correct parametric dependencies in the symbolic representation. The goal is to obtain points at which cuts can be computed in a second phase, which needs to parse the levels and stages at the right order.

2. Since $l=3$ does not receive any transfer from lower levels ($\tau_{\to 3}=0$), the last level triggers the updating procedure. The symbolic mechanism applied to $l=3$ will detect parametric dependencies in the floating cuts that are similar to those arising in the individualistic setting for $l=3$. The functions $\mathfrak{Q}_{l,r}^t(\cdot,\cdot)$ with $l=3$ are updated making a forward-backward pass at $l=3$, that fills the values for Z_3.

3. The next step is to update the functions $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$ for $l=2$. This is quite natural, since the iterates at $l=2$ would be more informed if they could take into account the implied costs of $l=3$. Accordingly, for all $t=1,\ldots,\mathtt{T}$ and $r=1,\ldots,S$, we compute a cut for the value function $U_{l+1,r}^t(x_{l+1}^{t-1}, x_{l,r}^t)$, where x_{l+1}^{t-1} and $x_{l,r}^t$ are taken from the tuples (Z_1, Z_2, Z_3). Precisely, x_{l+1}^{t-1} is the forward iterate associated with the scenario sampled at stage $t-1$ and $x_{l,r}^t$ is the corresponding value at Z_2 associated with scenario r. Note that there is a difference relative to the scenario used for x_{l+1}^{t-1} and $x_{l,r}^t$, as already announced. On the initialization step, we sampled a sequence of scenarios $s=(s_1,\ldots,s_{\mathtt{T}})$. The value used for x_{l+1}^{t-1} is the one associated with s_{t-1} and not with the r index. After such step, all functions $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$ for $l=2$ have been updated.

4. The next step is to run a forward-backward pass at $l=2$, but this time updating the future costs $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$, to detect the parametric dependencies from $l=3$ that impact on $l=2$. We start computing the cuts at the last stage, $\mathtt{T}$.

5. Since now $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$ for $l=2$ depends on x_{l+1}^{t-1}, after aggregating the symbolic cuts, we realize that the SDDP problem at level $l=2$ depends not only Z_1, but also on the forward decisions at $l=3$ represented by Z_3. However, for the algorithm this dependence is dealt with extremely easily. A particularity is that the instantaneous cost $\mathfrak{U}_{l,r}^t(\cdot,\cdot)$ for $l=2$ influences the future cost $\mathfrak{Q}_l^{t+1}(\cdot,\cdot)$ for $l=2$, which is again extremely natural.

6. The algorithm continues in the same fashion at $l = 1$. We start updating $\mathfrak{U}^t_{l,r}(\cdot,\cdot)$ for $l = 1$ and then perform a forward-backward pass at $l = 1$ updating the future costs and performing parameter detection. We again observe that SDDP problem at $l = 1$ depends on some components of Z_2 and Z_3.

7. When the forward-backward pass at $l = 1$ is finished, we sample a new sequence of scenarios $s = (s_1, \ldots, s_{\mathtt{T}})$ to make the initialization step again, without cut computation in the backward pass, and go back to the forward-backward at $l = 3$ again. In other words, we sample $s = (s_1, \ldots, s_{\mathtt{T}})$ and compute a new tuple (Z_1, Z_2, Z_3) without adding cuts. Then, we go back to step 2 and start all over with $l = 3$.

For each sample $s = (s_1, \ldots, s_{\mathtt{T}})$ obtained, we also obtain for each level a realization of an estimate of a lower bound and another one for the upper bound. After averaging these realizations of the upper and lower bounds, we obtain an expected gap, which is used to stop the algorithm.

Back to the best response setting, it is important to understand that every time some component of Z_2 and Z_3 change, the cuts available at $l = 1$ need to be transported ("floated"), so that they are still valid for the new values of Z_2 and Z_3. The symbolic cuts enable the application of a best-response iteration to our trilevel nested multi-stage stochastic setting.

4.3 Numerical Assessment

Our last set of experiments illustrates the nested multi-stage stochastic setting with the profit sharing mechanism, solved by the technique based on symbolic dependencies and floating cuts described in the previous sections. Since the procedure is quite involved, we use for comparison in the benchmark Algorithm 1 with $\tau_{3\to1} = 0$. This is the deterministic model with profit sharing; our rationale is that results should be similar if scenarios do not vary too much. In our runs, such variability depends on the standard deviation of considered scenarios shown in Figure 5.

The range chosen for the profit-sharing percentages is

$$\tau_{2\to1} = \tau_{3\to2} \in [0.05, 0.9]\,.$$

Our prototype code is not efficient and multi-stage stochastic trilevel problems are computationally heavy. To keep running times manageable for our code, the configuration from Section 3.3 is run with shorter time horizon and less scenarios, taking $\mathtt{T} := 8$ and $S := 5$. Each run takes 2 hours, ending with gaps smaller than 3% after 100 forward-backward iterations.

The simulation phase, with the system operating with the obtained policies uses 100 out-of-sample scenarios with the profile reported in Figure 5, truncated at $\mathtt{T} = 8$.

Figure 7 follows the premises in Figure 2, with vertical bars representing the total gain for each level. The results with the deterministic configurations is shown on the left, and the stochastic one on the right, taking the expected value of the cost of the 100 simulations. Numeric values for the transfers can be found in the Appendix A, Tables 4 and 5.

Near the time horizon $\mathtt{T} = 8$ prices in Figure 5 are high. This feature, combined with the short time that water has to travel downhill, increases the market power of the hydroplant at level 1. This phenomenon is perceptible when comparing the output on the left and right plots in Figure 7. In the deterministic case on the left, transferring 30% of the net margin is acceptable to the level in the top. By contrast, for the stochastic model, the fraction jumps $\tau_{2\to1} = \tau_{3\to2} \geq 0.7$. The owner in the top will accept not to play opportunistically and will stop withholding water only if the payment received from the lower level is at least 70% of its net margin. In a somewhat indirect manner, such significant difference gives a quantitative perception of the value of the stochastic solution in our nested trilevel setting, see [Bir82] and [Esc+07].

Concluding remarks

The issue of market power mitigation in multi-owned hydro cascades is among the main causes that hydro systems did not undergo the same privatization process that thermally dominated systems experienced

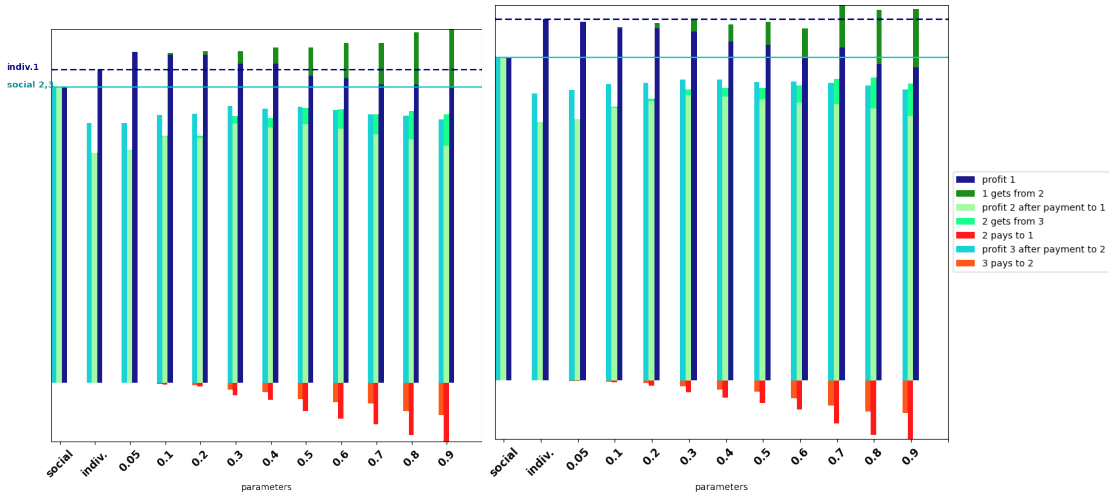

Figure 7: Wealth with deterministic and stochastic sharing policies (left and right). Since the latter considers few scenarios with little dispersion, the pattern of the bars is similar to the deterministic case. For level 1, any configuration of parameters above the dashed horizontal line is acceptable. For levels 2 and 3, the best is to get the closest to the solid horizontal line. In the stochastic model, the rightmost configurations that are satisfactory for the three levels are $\tau_{2\to1} = \tau_{3\to2} \geq 0.7$, whereas $\tau_{2\to1} = \tau_{3\to2} \geq 0.3$ suffices in the deterministic case. With the stochastic model and for $\tau_{2\to1}$ and $\tau_{3\to2}$ closer to 1, level 1 profit (the blue section of the bar) is close to the one obtained with the social profit (indicated by the solid horizontal line). With those configurations level 1 achieves a gain comparable to the individualistic policy (the dashed line) only after receiving a payment from level 2. For those configurations, the uphill level $l = 1$ behaves similarly to a confiscatory agent.

worldwide. While being a topic of high applied value for countries like Brazil and Canada, it also involves the practical solution of advanced game-theoretic models, some of which do not have effective solution strategies yet, see [FP03]. Our proposal provides a response in that direction, as it guarantees a more efficient management of the overall system. The methodology is shown to terminate finitely for the deterministic and two-stage settings. The approach can be applied to nested multi-stage stochastic programs thanks to the innovative concept of floating cuts defined symbolically in an SDDP framework.

Acknowledgments The first and second authors are grateful to Fondation Ecole Polytechnique, France, for the support through the 2018-2019 Gaspard Monge Visiting Professor Program. The third author is supported in part by CNPq Grant 303913/2019-3, by FAPERJ Grant E-26/202.540/2019, and by PRONEX–Optimization.

References

[ADCM91] I. Adler and R. D. C. Monteiro. “Limiting behavior of the affine scaling continuous trajectories for linear programming problems”. *Mathematical Programming* 50.1 (1991), pp. 29–51 (cit. on p. 6).

[Ber95] D. Bertsekas. *Nonlinear programming*. Belmont, Massachusetts: Athena Scientific, 1995 (cit. on p. 6).

[Bez+17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. “Julia: A fresh approach to numerical computing”. *SIAM Review* 59.1 (2017), pp. 65–98 (cit. on p. 9).

[Bir82] J. R. Birge. “The value of the stochastic solution in stochastic linear programs with fixed recourse”. *Mathematical Programming* 24.1 (1982), pp. 314–325 (cit. on p. 25).

[Ca+09] J. P. S. Catalão, S. J. P. S. Mariano, V. M. F. Mendes, and L. A. F. M. Ferreira. "Scheduling of Head-Sensitive Cascaded Hydro Systems: A Nonlinear Approach". *IEEE Transactions on Power Systems* 24.1 (2009), pp. 337–346 (cit. on p. 3).

[CA13] F. Cicconet and K. C. Almeida. "Decentralized dispatch with price consistency in predominantly hydro systems". *IEEE Grenoble Conference* (2013), pp. 1–6 (cit. on p. 2).

[CaPM10] J. Catalão, H. Pousinho, and V. Mendes. "Scheduling of head-dependent cascaded hydro systems: Mixed-integer quadratic programming approach". *Energy Conversion and Management* 51.3 (2010), pp. 524–530 (cit. on p. 3).

[CLY18] L. Chang L. Li, P. Liu, and B. Yu. "Multi-Owner Scheduling for Cascade Hydro Power Using Multi-Objective Optimization Technique". *Eighth International Conference on Information Science and Technology* (2018), pp. 194–199 (cit. on p. 3).

[Dem02] S. Dempe. *Foundations of Bilevel Programming*. Springer, US, 2002 (cit. on p. 6).

[DSS09] A. Daniilidis, C. Sagastizábal, and M. Solodov. "Identifying Structure of Nonsmooth Convex Functions by the Bundle Technique". *SIAM Journal on Optimization* 20 (2009), pp. 820–840 (cit. on p. 6).

[Esc+07] L. F. Escudero, A. Garín, M. Merino, and G. Pérez. "The value of the stochastic solution in multistage problems". *TOP* 15.1 (2007), pp. 48–64 (cit. on p. 25).

[FP03] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems, volumes 1 and 2*. Springer Series in Operations Research and Financial Engineering, 2003 (cit. on p. 26).

[GB21] V. Guigues and M. Bandarra. "Single cut and multicut SDDP with cut selection for multistage stochastic linear programs: convergence proof and numerical experiments". *Computational Management Science* 18.1 (2021), pp. 125–148 (cit. on p. 8).

[GMH10] A. Gjelsvik, B. Mo, and A. Haugstad. "Long- and Medium-term Operations Planning and Stochastic Modelling in Hydro-dominated Power Systems Based on Stochastic Dual Dynamic Programming". *Handbook of Power Systems* (2010), pp. 33–35 (cit. on p. 2).

[GR12] V. Guigues and W. Romisch. "Sampling-based decomposition methods for multistage stochastic programs based on extended polyhedral risk measures". *SIAM J. Optim.* 22.1 (2012), pp. 286–312 (cit. on pp. 8, 17).

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I and II*. Springer Berlin Heidelberg, 1993 (cit. on p. 4).

[Int18] International Hydropower Association. *The World's Water Battery: Pumped Hydropower Storage and the Clean Energy Transition*. 2018 (cit. on p. 1).

[IRE20] IRENA. "Renewable Energy Highlights". `irena.org/publications/2020/Jul/Renewable-energy-statistics-2020` (2020) (cit. on p. 1).

[Kel99] R. Kelman. "Competitive Schemes in Hydrothermal Systems: Economic Efficiency and Strategic Behaviour". *M.Sc Thesis, COPPE/UFRJ* (1999) (cit. on p. 2).

[KLBP01] R. Kelman, A. N. L. Barroso, and M. V. F. Pereira. "Market power assessment and mitigation in hydrothermal systems". *IEEE Transactions on Power Systems* 16.3 (2001), pp. 354–359 (cit. on p. 2).

[KM14] V. Kozmík and D. P. Morton. "Evaluating policies in risk-averse multi-stage stochastic programming". *Mathematical Programming* 152.1–2 (2014), pp. 275–300 (cit. on p. 17).

[LQ18] F.-F. Li and J. Qiu. "Multi-Objective Reservoir Optimization Balancing Energy Generation and Firm Power". *Energies* 8.1 (2018), pp. 6962–6976 (cit. on p. 3).

[MM20] R. Moita and D. Monte. "Hydroelectric Generators Competing in Cascades". *Revista Brasileira de Economia* 74.1 (2020) (cit. on p. 2).

[OSS11] W. Oliveira, C. Sagastizábal, and S. Scheimberg. "Inexact Bundle Methods for Two-Stage Stochastic Programming". *SIAM Journal on Optimization* 21.2 (2011), pp. 517–544 (cit. on p. 6).

[PP91] M. Pereira and L. Pinto. "Multi-stage stochastic optimization applied to energy planning". *Mathematical Programming* 52.1 (1991), pp. 359–375 (cit. on pp. 3, 14).

[Sha11] A. Shapiro. "Analysis of stochastic dual dynamic programming method". *European Journal of Operational Research* 209.1 (2011), pp. 63–72 (cit. on pp. 8, 16, 17).

[SMK18] B. Swenson, R. Murray, and S. Kar. "On Best Response Dynamics in Potential Games". *SIAM Journal on Control and Optimization* 56.4 (2018), pp. 2734–2767 (cit. on p. 23).

[TD17] 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 (2017), pp. 57–79 (cit. on p. 2).

[VW69] R. Van Slyke and R. Wets. "L-shaped linear programs with applications to control and stochastic programming". *SIAM Journal on Applied Mathematics* 17 (1969), pp. 638–663 (cit. on pp. 3, 14).

A Appendix with tables

Table 3: Deterministic run, Algorithm 1, values in Figure 2.

Policy Type	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$	Tot. Profit
Social	332.17	439.23	718.04	1489.45
Individual	+4.18%	-5.97%	-3.73%	-2.62%
$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.02$	+4.18%	-5.97%	-3.73%	-2.62%
$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.05$	+4.18%	-5.59%	-3.70%	-2.50%
$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.10$	+4.40%	-3.35%	-2.81%	-1.36%
$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.20$	+5.32%	-1.66%	-2.40%	-0.04%
$\tau_{2\to1} = \tau_{3\to1} = \tau_{3\to2} = 0.40$	+8.36%	-2.00%	-3.21%	-0.02%
$\tau_{2\to1} = \tau_{3\to2} = 0.20$, $\tau_{3\to1} = 0.00$	+4.58%	-3.48%	-2.68%	-1.29%
$\tau_{2\to1} = \tau_{3\to2} = 0.40$, $\tau_{3\to1} = 0.00$	+6.19%	-2.19%	-2.36%	-0.04%

Table 5: This table continues the analysis of Table 4. It shows the percentage variations of profits relative to the social policy after the profit-sharing payments. Recall that the payments are based on what exceeds the individualistic profits. It is interesting to note the results for $\tau_{2\to1}$ and $\tau_{3\to2}$ closer to 1. The profits before profit-sharing payments are closer to social profits and after the payments are closer to the individualistic profits. In these cases, hydro $l = 1$ acts analogously to a confiscatory agent.

	Deterministic			Stochastic (Expected Value)		
Policy Type	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$
Individual	+12 %	-22 %	-12 %	+12 %	-20 %	-11 %
$\tau_{2\to1} = \tau_{3\to2} = 0.05$	+12 %	-21 %	-12 %	+11 %	-19 %	-10 %
$\tau_{2\to1} = \tau_{3\to2} = 0.10$	+12 %	-16 %	-09 %	+09 %	-15 %	-08 %
$\tau_{2\to1} = \tau_{3\to2} = 0.20$	+13 %	-16 %	-09 %	+11 %	-13 %	-08 %
$\tau_{2\to1} = \tau_{3\to2} = 0.30$	+15 %	-09 %	-07 %	+13 %	-10 %	-07 %
$\tau_{2\to1} = \tau_{3\to2} = 0.40$	+19 %	-10 %	-07 %	+14 %	-10 %	-07 %
$\tau_{2\to1} = \tau_{3\to2} = 0.50$	+22 %	-08 %	-07 %	+17 %	-10 %	-07 %
$\tau_{2\to1} = \tau_{3\to2} = 0.60$	+27 %	-10 %	-08 %	+19 %	-11 %	-07 %
$\tau_{2\to1} = \tau_{3\to2} = 0.70$	+31 %	-12 %	-09 %	+32 %	-11 %	-08 %
$\tau_{2\to1} = \tau_{3\to2} = 0.80$	+40 %	-14 %	-10 %	+35 %	-13 %	-09 %
$\tau_{2\to1} = \tau_{3\to2} = 0.90$	+47 %	-18 %	-11 %	+39 %	-17 %	-10 %

Table 4: Stochastic run with profit sharing, values in Figure 7. The values reported are raw profits before any transfer. Percentage variations of the profits after transfer are reported in Table 5.

	Deterministic			Stochastic (Expected Value)		
Policy Type	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$	Profit $l = 1$	Profit $l = 2$	Profit $l = 3$
Social	108.31	146.24	230.76	109.87	143.40	227.20
Individual	121.50	114.74	202.67	123.12	114.40	202.43
$\tau_{2\to1} = \tau_{3\to2} = 0.05$	121.48	115.21	202.67	122.25	116.80	205.52
$\tau_{2\to1} = \tau_{3\to2} = 0.10$	120.74	123.07	209.98	119.33	121.81	208.93
$\tau_{2\to1} = \tau_{3\to2} = 0.20$	120.74	122.78	211.76	119.40	126.09	211.32
$\tau_{2\to1} = \tau_{3\to2} = 0.30$	117.29	135.11	221.22	118.34	131.47	215.65
$\tau_{2\to1} = \tau_{3\to2} = 0.40$	117.30	135.11	221.22	115.87	133.04	217.70
$\tau_{2\to1} = \tau_{3\to2} = 0.50$	113.15	141.27	227.38	114.54	134.41	219.19
$\tau_{2\to1} = \tau_{3\to2} = 0.60$	111.68	143.92	227.38	109.61	136.78	222.52
$\tau_{2\to1} = \tau_{3\to2} = 0.70$	109.87	143.40	227.20	112.99	142.60	226.89
$\tau_{2\to1} = \tau_{3\to2} = 0.80$	109.06	146.24	229.70	107.68	144.28	228.38
$\tau_{2\to1} = \tau_{3\to2} = 0.90$	108.31	146.24	230.76	106.73	143.14	227.76

B Stochastic Algorithm with Cost Sharing

We now give the precise description of the stochastic cost sharing method. The notation is the same of Section 4, except that the forward iterates associated with a forward path of scenarios $s = (s_1, \ldots, s_T)$ is denoted as by $\hat{Z}_l^t = (\hat{x}_{l,s}^t, \ldots, \hat{x}_{l,s}^{(\mathtt{T}-1)})$, where $\hat{x}_{l,s}^t = \hat{x}_{lw}^t$, $w = s_t$, and with $\hat{Z}_l^{\mathtt{T}}$ an empty vector. The sequence of Algorithms 4 to 9 matches the sequence of steps described in Section 4. The name of those steps related to a forward-backward SDDP pass is shortened to FB Pass, for convenience.

Note that at the initialization step of Algorithm 4 we set $\hat{x}_{l+1,\omega}^{t-1,k} = 0$ and $\hat{x}_{l+2,\omega}^{t-1,k} = 0$. The choice of the value "zero" is irrelevant, since it is just the first trial point for the approximations of the value functions and the value functions are initialized with $-M$. Therefore, any initial value for $\hat{x}_{l+1,\omega}^{t-1,k}$ and $\hat{x}_{l+2,\omega}^{t-1,k}$ gives the same results.

It is extremely important to notice that ideally, the algorithm would have to be able to generate cutting plane approximations for the value functions at different levels, such that the cuts are tight are the Nash triples. However, computing the Nash triples is hard. For this reason, we calibrate the value functions at the best-response iterates.

Algorithm 4 STOCHASTIC ALGORITHM WITH COST SHARING (PART 1)

Initialization. Take $\varepsilon \geq 0$ and a sufficiently large constant $M > 0$. Set $k = 1$. For $l = 1, 2, 3$, and $t \geq 2$, let $Q_{l,s}^{t,k} \equiv -M$ and $U_{l,s}^{t,k} \equiv -M$. For $l = 1$ and $k = 1$, set $\hat{x}_{l+1,\omega}^{t-1,k} = 0$ and $\hat{x}_{l+2,\omega}^{t-1,k} = 0$ for all t, ω, where ω refers to the scenario.

Step 1: Sampling. Obtain a sample $s_t^k \in \{1, \ldots, S\}$ for each $t = 1, \ldots, \mathtt{T}$.

Step 2: Get Feasible Iterates at $l = 1$. For each $t = 1, \ldots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,s}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,\omega}^{t-1,k})\,. \end{cases}$$

For each $t = \mathtt{T}, \ldots, 2$ and for each $r = 1, \ldots, S$, take $\omega = s_{t-1}^k$ and compute

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k})\,. \end{cases}$$

Step 3: Get Feasible Iterates at $l = 2$. For each $t = 1, \ldots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,s}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,s}^{tk})\,. \end{cases}$$

For each $t = \mathtt{T}, \ldots, 2$ and for each $r = 1, \ldots, S$, take $\omega = s_{t-1}^k$ and compute

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,r}^{tk})\,. \end{cases}$$

Algorithm 5 STOCHASTIC ALGORITHM WITH COST SHARING (PART 2)

Step 4: FB Pass at $l = 3$. For each $t = 1, \ldots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,s}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,s}^{tk})\,. \end{cases}$$

Step 4.1. Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

Step 4.2. For $t = \mathtt{T}, \ldots, 2$.

Step 4.2.1: Cut computation. For $r = 1, \ldots, S$, compute

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,r}^{tk})\,. \end{cases}$$

Obtain subgradients such that for all $x_l^{t-1}, x_{l-1,r}^t$ and Z_{l-1}^{t+1} the value function $Q_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t, Z_{l-1}^{t+1})$ lies above

$$\begin{array}{lllll} Q_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,r}^{t,k}, Z_{l-1}^{t+1,k}) & + & (\lambda_l^{t-1,k})^\top(x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) & + \\ (\mu_{l-1,r}^{t,k})^\top(x_{l-1,r}^t - x_{l-1,r}^{t,k}) & + & (\nu_{l-1}^{t+1,k})^\top(Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}). & \end{array} \tag{25}$$

Step 4.2.1: Cut Aggregation. Average the cuts in (25) to obtain a cut such that $\mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, Z_{l-1}^t)$ lies above

$$\mathfrak{Q}_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, Z_{l-1}^{tk}) + (\phi_l^{t-1,k})^\top(x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) + (\rho_{l-1}^{tk})^\top(Z_{l-1}^t - Z_{l-1}^{tk}). \tag{26}$$

Define $\mathfrak{Q}_{l,r}^{t,k+1}$ as a maximum between $\mathfrak{Q}_{l,r}^{t,k}$ and (26).

Step 4.3: Calculation of Bounds. Take $\overline{u}_l^k = \sum_t f_{l,s}^t \hat{x}_{l,s}^t$ where $s = s_t^k$. Take $\underline{u}_l^k$ as the optimal value of the first state problem after the backward step (Step 4.2).

Algorithm 6 STOCHASTIC ALGORITHM WITH COST SHARING (PART 3)

Step 5: Cost-Sharing to $l = 2$. For each stage $t = 1, \ldots, \mathtt{T}$ and each scenario $s = 1, \ldots, S$, take $\omega = s_{t-1}^k$ and compute subgradients

$$(\alpha_{lts}^k, \beta_{lts}^k) \in \partial U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, x_{l,s}^{tk}).$$

Then, take $U_{l+1,s}^{t,k+1}$ as a maximum between $U_{l+1,s}^{t,k}$ and the affine function

$$U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, x_{l,s}^{tk}) + (\alpha_{lts}^k)^\top(x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) + (\beta_{lts}^k)^\top(x_{l,s}^t - x_{l,s}^{tk}).$$

Note that the variables are x_{l+1}^{t-1} and $x_{l,s}^t$, which represent, respectively the forward decision at stage $t-1$ at level 3 and the decision taken at scenario s and stage t at level 2.

Algorithm 7 STOCHASTIC ALGORITHM WITH COST SHARING (PART 4)

Step 6: FB Pass at $l = 2$. For each $t = 1, \ldots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & f_{l,s}^{t\ \top} y + \mathfrak{Q}_l^{t+1,k}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,s}^{tk})\,. \end{cases}$$

Step 6.1. Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

Step 6.2. For $t = \mathtt{T}, \ldots, 2$.

Step 6.2.1: Cut computation. For $r = 1, \ldots, S$, compute

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & f_{l,r}^{t\ \top} y + \mathfrak{Q}_l^{t+1,k+1}(y, Z_{l-1}^{t+1,k}, \hat{Z}_{l+1}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,r}^{tk})\,. \end{cases}$$

Obtain subgradients such that for all $x_l^{t-1}, x_{l-1,r}^t, x_{l+1}^{t-1}, \hat{Z}_{l+1}^t$ and Z_{l-1}^{t+1} the value function $Q_{l,r}^t(x_l^{t-1}, x_{l-1,r}^t, x_{l+1}^{t-1}, \hat{Z}_{l+1}^t, Z_{l-1}^{t+1})$ lies above

$$\begin{array}{lllll} Q_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, x_{l-1,r}^{t,k}, \hat{x}_{l+1,\omega}^{t-1}, \hat{Z}_{l+1}^{t,k}, Z_{l-1}^{t+1,k}) & + & (\lambda_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) & + & \\ (\mu_{l-1,r}^{t,k})^\top (x_{l-1,r}^t - x_{l-1,r}^{tk}) & + & (\nu_{l-1}^{t+1,k})^\top (Z_{l-1}^{t+1} - Z_{l-1}^{t+1,k}) & + & \\ (\xi_{l+1,r}^{t,k})^\top (x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) & + & (\pi_{l+1}^{t+1,k})^\top (\hat{Z}_{l+1}^t - \hat{Z}_{l+1}^{t,k}). & & \end{array} \tag{27}$$

Step 6.2.1: Cut Aggregation. Average the cuts in (27) to obtain a cut such that $\mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, Z_{l-1}^t, \hat{Z}_{l+1}^{t-1})$ lies above

$$\begin{array}{lllll} \mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, Z_{l-1}^{tk}, \hat{Z}_{l+1}^{t,k}) & + & (\phi_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) & + & \\ (\rho_{l-1}^{tk})^\top (Z_{l-1}^t - Z_{l-1}^{tk}) & + & (\psi_{l+1}^{tk})^\top (\hat{Z}_{l+1}^{t-1} - \hat{Z}_{l+1}^{t-1,k}). & & \end{array} \tag{28}$$

Define $\mathfrak{Q}_{l,r}^{t,k+1}$ as a maximum between $\mathfrak{Q}_{l,r}^{t,k}$ and (28).

Step 6.3: Calculation of Bounds. Take $\overline{u}_l^k = \sum_t f_{l,s}^t \hat{x}_{l,s}^t + \tau_{\to l} \overline{u}_{l+1}^k$ where $s = s_t^k$. Take $\underline{u}_l^k$ as the optimal value of the first state problem after the backward step (Step 6.2).

Algorithm 8 STOCHASTIC ALGORITHM WITH COST SHARING (PART 5)

Step 7: Cost-Sharing to $l = 1$. For each stage $t = 1, \ldots, \mathtt{T}$ and each scenario $s = 1, \ldots, S$, take $\omega = s_{t-1}^k$ and compute subgradients

$$(\alpha_{lts}^k, \beta_{lts}^k, \gamma_{lts}^k) \in \partial U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, x_{l,s}^{tk}).$$

Then, take $U_{l+1,s}^{t,k+1}$ as a maximum between $U_{l+1,s}^{t,k}$ and the affine function

$$U_{l+1,s}^t(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, x_{l,s}^{tk}) + (\alpha_{lts}^k)^\top (x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) + (\beta_{lts}^k)^\top (x_{l+2}^{t-1} - \hat{x}_{l+2,\omega}^{t-1,k}) + (\gamma_{lts}^k)^\top (x_{l,s}^t - x_{l,s}^{tk}).$$

Algorithm 9 STOCHASTIC ALGORITHM WITH COST SHARING (PART 6)

Step 8: FB Pass at $l = 1$. For each $t = 1, \ldots, \mathtt{T}$, take $s = s_t^k$ and $\omega = s_{t-1}^k$, and compute

$$\hat{x}_{l,s}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,s}^t}^\top y + \mathfrak{Q}_l^{t+1,k}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,s}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,s}^t(\hat{x}_{l,\omega}^{t-1,k})\,. \end{cases}$$

Step 8.1. Take $\mathfrak{Q}_l^{\mathtt{T}+1,k+1} = 0$.

Step 8.2. For $t = \mathtt{T}, \ldots, 2$.

Step 8.2.1: Cut computation. For $r = 1, \ldots, S$, compute

$$x_{l,r}^{tk} \quad \text{solving} \quad \begin{cases} \min & {f_{l,r}^t}^\top y + \mathfrak{Q}_l^{t+1,k+1}(y, \hat{Z}_{l+1}^{t,k}, \hat{Z}_{l+2}^{t,k}) + \tau_{\to l} U_{l+1,r}^{t,k+1}(\hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}, y) \\ \text{s.t.} & y \in F_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k})\,. \end{cases}$$

Obtain subgradients such that for all $x_l^{t-1}, \hat{Z}_{l+1}^t, \hat{Z}_{l+2}^t$ and $x_{l+1}^{t-1}, \hat{x}_{l+2}^{t-1}$ the value function $Q_{l,r}^t(x_l^{t-1}, \hat{Z}_{l+1}^t, \hat{Z}_{l+2}^t, x_{l+1}^{t-1}, x_{l+2}^{t-1})$ lies above

$$\begin{array}{lll} Q_{l,r}^t(\hat{x}_{l,\omega}^{t-1,k}, \hat{Z}_{l+1}^{tk}, \hat{Z}_{l+2}^{t,k}, \hat{x}_{l+1,\omega}^{t-1,k}, \hat{x}_{l+2,\omega}^{t-1,k}) & + & (\lambda_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) \quad + \\ (\mu_{l+1,r}^{t,k})^\top (x_{l+1}^{t-1} - \hat{x}_{l+1,\omega}^{t-1,k}) & + & (\nu_{l+1}^{t+1,k})^\top (\hat{Z}_{l+1}^t - \hat{Z}_{l+1}^{t,k}) \quad + \\ (\xi_{l+2,r}^{t,k})^\top (x_{l+2}^{t-1} - \hat{x}_{l+2,\omega}^{t-1,k}) & + & (\pi_{l+2}^{t+1,k})^\top (\hat{Z}_{l+2}^t - \hat{Z}_{l+2}^{t,k}). \end{array} \tag{29}$$

Step 8.2.1: Cut Aggregation. Average the cuts in (29) to obtain a cut such that $\mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ lies above

$$\begin{array}{lll} \mathfrak{Q}_{l,r}^t(x_l^{t-1,k}, \hat{Z}_{l+1}^{tk}, \hat{Z}_{l+2}^{t,k}) & + & (\phi_l^{t-1,k})^\top (x_l^{t-1} - \hat{x}_{l,\omega}^{t-1,k}) \quad + \\ (\rho_{l+1}^{tk})^\top (\hat{Z}_{l+1}^{t-1} - Z_{l+1}^{tk}) & + & (\psi_{l+2}^{tk})^\top (\hat{Z}_{l+2}^{t-1} - \hat{Z}_{l+2}^{t-1,k}). \end{array} \tag{30}$$

Take $\mathfrak{Q}_{l,r}^{t,k+1}(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ as a maximum between $\mathfrak{Q}_{l,r}^{t,k}(x_l^{t-1,k}, \hat{Z}_{l+1}^{t-1}, \hat{Z}_{l+2}^{t-1})$ and (30).

Step 8.3: Calculation of Bounds. Take $\overline{u}_l^k = \sum_t f_{l,s}^t \hat{x}_{l,s}^t + \tau_{\to l} \overline{u}_{l+1}^k$ where $s = s_t^k$. Take $\underline{u}_l^k$ as the optimal value of the first state problem after the backward step (Step 8.2).

Step 9: Stopping Test. Stop if for all $l = 1, 2, 3$ the average of $\overline{u}_l^k$ and $\underline{u}_l^k$ across k are close enough or the lower bounds $\underline{u}_l^k$ stabilized. Else, set $k = k + 1$ and go back to Step 1.