SIAM J. OPTIM.
Vol. 30, No. 2, pp. 1223–1250

EXACT CONVERGING BOUNDS FOR STOCHASTIC DUAL DYNAMIC PROGRAMMING VIA FENCHEL DUALITY*

VINCENT LECLÈRE†, PIERRE CARPENTIER‡, JEAN-PHILIPPE CHANCELIER†, ARNAUD LENOIR§, AND FRANÇOIS PACAUD¶

Abstract. The stochastic dual dynamic programming (SDDP) algorithm has become one of the main tools used to address convex multistage stochastic optimal control problems. Recently a large amount of work has been devoted to improving the convergence speed of the algorithm through cut selection and regularization, and to extending the field of applications to nonlinear, integer, or risk-averse problems. However, one of the main downsides of the algorithm remains the difficulty in giving an upper bound of the optimal value, usually estimated through Monte Carlo methods and therefore difficult to use in the stopping criterion of the algorithm. In this paper we present a dual SDDP algorithm that yields a converging exact upper bound for the optimal value of the optimization problem. As an easy consequence of our approach, we show how to compute an alternative control policy based on an inner approximation of Bellman value functions instead of the outer approximation given by the standard SDDP algorithm. We illustrate the approach on an energy production problem involving zones of production and transportation links between the zones. The numerical experiments we carry out on this example show the effectiveness of the method.

Key words. stochastic programming, dynamic programming, SDDP, Fenchel conjugacy

AMS subject classifications. 90C15, 90C39, 49N15

DOI. 10.1137/19M1258876

1. Introduction.

In this paper, we consider a risk neutral multistage stochastic optimization problem with continuous decision variables. We adopt the stochastic optimal control point of view; that is, we work with explicit control and state variables in order to deal with an explicit dynamics of the system.

1.1. Stochastic optimization problem in discrete time.

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space, where Ω is the set of possible outcomes, $\mathcal{A}$ the associated σ-field, and $\mathbb{P}$ the probability measure. We denote by $[\![0,T]\!]$ the discrete time span $\{0,1,\dots,T\}$, and we define upon it three processes, $\boldsymbol{X} = \left\{\boldsymbol{X}_t\right\}_{t\in[\![0,T]\!]}$, $\boldsymbol{U} = \left\{\boldsymbol{U}_t\right\}_{t\in[\![1,T]\!]}$, and $\boldsymbol{\xi} = \left\{\boldsymbol{\xi}_t\right\}_{t\in[\![1,T]\!]}$, where for all t, $\boldsymbol{X}_t : \Omega \to \mathbb{R}^{n_x}$, $\boldsymbol{U}_t : \Omega \to \mathbb{R}^{n_u}$, and $\boldsymbol{\xi}_t : \Omega \to \mathbb{R}^{n_\xi}$ are random variables representing, respectively, the state, the control, and the noise variables. The state process $\boldsymbol{X}$ is assumed to follow the linear dynamics

$$\boldsymbol{X}_0 = x_0, \quad \boldsymbol{X}_{t+1} = A_t\boldsymbol{X}_t + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1} \qquad \forall t \in [\![0, T-1]\!]\,,$$

where x_0 is the given initial state at time 0 and where $A_t \in \mathbb{R}^{n_x\times n_x}$, $B_{t+1} \in \mathbb{R}^{n_x\times n_u}$, and $C_{t+1} \in \mathbb{R}^{n_x\times n_\xi}$ are given deterministic matrices. Moreover we assume that both control and state variables are subject to bound constraints, that is, $\underline{u}_{t+1} \le \boldsymbol{U}_{t+1} \le$

*Received by the editors April 29, 2019; accepted for publication (in revised form) February 24, 2020; published electronically April 28, 2020.
https://doi.org/10.1137/19M1258876
Funding: This work was supported by the FMJH "Program Gaspard Monge for Optimization and Operations Research and Their Interactions with Data Science" and from the support of EDF.
†CERMICS, Ecole des Ponts, Marne-la-Vallée, France (vincent.leclere@enpc.fr, chancelier@cermics.enpc.fr).
‡UMA, ENSTA Paris, Institut Polytechnique de Paris, Paris, France (pierre.carpentier@ensta-paris.fr).
§EDF, 91120 Palaiseau, France (arnaud.lenoir@edf.fr).
¶CERMICS, Ecole des Ponts, Efficacity, Marne-la-Vallée, France (francois.pacaud@enpc.fr).

$\overline{u}_{t+1}$ and $\underline{x}_{t+1} \leq \boldsymbol{X}_{t+1} \leq \overline{x}_{t+1}$ for all $t \in [\![0, T-1]\!]$, and satisfy a linear coupling constraint $D_t \boldsymbol{X}_t + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \leq 0$, where $D_t \in \mathbb{R}^{n_c \times n_x}$, $E_{t+1} \in \mathbb{R}^{n_c \times n_u}$, and $G_{t+1} \in \mathbb{R}^{n_c \times n_\xi}$ are given deterministic matrices.

We assume that the problem has a hazard-decision[1] information structure; that is, the decision at time t is taken knowing the noise that affects the system between t and $t+1$. Accordingly, the decision $\boldsymbol{U}_{t+1}$ is a function of the uncertainties up to time $t+1$, which means that $\boldsymbol{U}_{t+1}$ has to be measurable with respect to the σ-field $\mathcal{F}_{t+1}$ generated by the uncertainties $(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{t+1})$. We write this *nonanticipativity constraint* as $\boldsymbol{U}_{t+1} \preceq \mathcal{F}_{t+1}$ for all $t \in [\![0, T-1]\!]$.

Finally, the cost incurred at each time $t \in [\![0, T-1]\!]$ is a linear function $a_t^\top \boldsymbol{X}_t + b_{t+1}^\top \boldsymbol{U}_{t+1}$ with $a_t \in \mathbb{R}^{n_x}$ and $b_{t+1} \in \mathbb{R}^{n_u}$, and the cost incurred at the final time T is $K(\boldsymbol{X}_T)$, where K is a polyhedral, hence convex lower semicontinuous, function. Note that the results obtained in this paper for a polyhedral final cost function can be adapted to the case where K is a convex lower semicontinuous function, Lipschitz-continuous on its domain.

Gathering all these elements, we get the following stochastic optimization problem:

$$\min_{\boldsymbol{X},\boldsymbol{U}} \quad \mathbb{E}\left[\sum_{t=0}^{T-1}(a_t^\top \boldsymbol{X}_t + b_{t+1}^\top \boldsymbol{U}_{t+1}) + K(\boldsymbol{X}_T)\right] \tag{1.1a}$$

$$\text{s.t.} \quad \boldsymbol{X}_0 = x_0,\ \boldsymbol{X}_{t+1} = A_t \boldsymbol{X}_t + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1} \quad \forall t \in [\![0, T-1]\!]\,, \tag{1.1b}$$

$$\underline{u}_{t+1} \leq \boldsymbol{U}_{t+1} \leq \overline{u}_{t+1},\ \underline{x}_{t+1} \leq \boldsymbol{X}_{t+1} \leq \overline{x}_{t+1} \quad \forall t \in [\![0, T-1]\!]\,, \tag{1.1c}$$

$$D_t \boldsymbol{X}_t + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \leq 0 \quad \forall t \in [\![0, T-1]\!]\,, \tag{1.1d}$$

$$\boldsymbol{U}_{t+1} \preceq \mathcal{F}_{t+1} \quad \forall t \in [\![0, T-1]\!]\,. \tag{1.1e}$$

We make the following assumption throughout the paper.

Assumption 1.1 (discrete white noise). The noise sequence $\{\boldsymbol{\xi}_t\}_{t \in [\![1,T]\!]}$ is assumed to be a sequence of independent random variables with finite support.

As is well known, independence is of paramount importance in obtaining dynamic programming equations, while finiteness of the support is required both to be able to compute exactly the expectation and for theoretical convergence reasons.

1.2. Stochastic dual dynamic programming and its weaknesses. Thanks to white noise Assumption 1.1, we can solve problem (1.1) by the dynamic programming approach (see the two reference books [3] and [4] for further details). This approach leads to the so-called Bellman value functions V_t, where $V_t(x)$ is the optimal value of the problem when starting at time t with state $\boldsymbol{X}_t = x$. These functions are obtained by solving the following recursive Bellman equation:

$$V_T(x) = K(x)\,, \tag{1.2a}$$

$$\hat{V}_t(x, \xi_{t+1}) = \inf_{u_{t+1}, x_{t+1}} a_t^\top x + b_{t+1}^\top u_{t+1} + V_{t+1}(x_{t+1}) \tag{1.2b}$$

$$\text{s.t.} \quad x_{t+1} = A_t x + B_{t+1}u_{t+1} + C_{t+1}\xi_{t+1}\,, \tag{1.2c}$$

$$\underline{u}_{t+1} \leq u_{t+1} \leq \overline{u}_{t+1}, \quad \underline{x}_{t+1} \leq x_{t+1} \leq \overline{x}_{t+1}\,, \tag{1.2d}$$

$$D_t x + E_{t+1}u_{t+1} + G_{t+1}\xi_{t+1} \leq 0\,, \tag{1.2e}$$

$$V_t(x) = \mathbb{E}\big[\hat{V}_t(x, \boldsymbol{\xi}_{t+1})\big]\,. \tag{1.2f}$$

[1] "Wait-and-see" in stochastic programming terminology.

When the state variable takes a finite number of possible values, we can solve the Bellman equation by exhaustive exploration of the state, yielding the exact solution of the problem. In the continuous linear-convex case, we can rely on polyhedral approximations. At each iteration k, the stochastic dual dynamic programming (SDDP) algorithm builds polyhedral approximations $\underline{V}_t^k$ of the functions V_t by using a sampled nested Benders decomposition (see [18, 10, 11, 12, 13] for the convergence of this approach). The polyhedral value functions $\underline{V}_t^k$ computed by SDDP are *outer approximations* of the functions V_t at each stage, that is, $\underline{V}_t^k \leq V_t$, so that the value $\underline{v}_0 = \underline{V}_0^k(x_0)$ is a true[2] lower bound on the optimal value $V_0(x_0)$ of problem (1.1).

Note that by providing a sequence of approximate value functions $\{\tilde{V}_t\}_{t\in[\![1,T]\!]}$, we are able to define an admissible strategy for problem (1.1). Indeed, solving problem (1.2b)–(1.2e) when replacing V_{t+1} by its approximation $\tilde{V}_{t+1}$ gives the control to be applied for any initial condition (x, ξ). Thus, functions $\underline{V}_t^k$ can be used to derive an admissible strategy, whose associated expected cost $\overline{v}_0^k$ gives an upper bound of the optimization problem value. Unfortunately, computing the expectation is usually out of reach, and we need to rely on some approximate computation. The most common way to perform that task is based on the Monte Carlo approach: it consists of simulating the control strategy induced by functions $\underline{V}_t^k$ along a (large) number M of noise scenarios, and then computing the arithmetic mean $\widehat{v}_0^M$ of the scenarios cost and the associated empirical standard deviation $\widehat{\sigma}_0^M$. The value $\widehat{v}_0^M$ is an approximate (statistical) upper bound of the optimal value of problem (1.1). Moreover, it is easy to obtain an asymptotic α-confidence interval $[\widehat{v}_0^M - z_\alpha \widehat{\sigma}_0^M, \widehat{v}_0^M + z_\alpha \widehat{\sigma}_0^M]$. Here $1-\alpha \in [0, 1]$ is a chosen confidence level and $z_\alpha = \Phi^{-1}(1-\alpha)$, Φ being the cumulative distribution function of the standard normal distribution.

The classical way to use this statistical upper bound in SDDP, as presented in [16], consists in testing at each iteration of the algorithm whether the available exact lower bound $\underline{v}_0$ is greater than the α-confidence lower bound $\widehat{v}_0^M - z_\alpha \widehat{\sigma}_0^M$, and to stop the algorithm in that case. Such a stopping criterion raises at least two difficulties: the Monte Carlo simulation increases the computational burden of SDDP, and the stopping test does not give any guarantee of convergence of the algorithm.

The first difficulty can be tackled by parallelizing the M simulations involved in the evaluation of the upper bound, and also by calculating the empirical mean $\widehat{v}_0$ over the last $\overline{k}$ iterations of the algorithm, thus enlarging the sample size from M to $\overline{k}M$ without additional computation (see [23, section 3.2]). The second difficulty induced by this stopping criterion was analyzed in [21]: the larger the standard deviation $\widehat{\sigma}_0$ and the confidence $(1-\alpha)$ are, the sooner the algorithm will be stopped. The author proposes another criterion based on the difference between the α-confidence upper bound $\widehat{v}_0 + z_\alpha \widehat{\sigma}_0$ and the exact lower bound $\underline{v}_0$ up to a prescribed accuracy level ε. Note that this stopping test is not necessarily convergent, in the sense that the stopping criterion might not be met in finite time, for example, if $\varepsilon < z_\alpha \widehat{\sigma}_0$. An interesting view on the class of stopping criteria in terms of statistical hypothesis tests was given in [14]; the authors compare different hypothesis tests of optimality[3] and so they find the stopping criteria proposed by [16, 21], as well as another one which ensures an upper bound on the probability of incorrectly claiming convergence (type II error). Moreover, the simulation scenarios are obtained using quasi–Monte Carlo or Latin hypercube sampling rather than raw Monte Carlo, so that the accuracy of the upper bound is increased. Nevertheless, all these stopping criteria are based on

[2] In particular this lower bound does not rely on a statistical approach.

[3] Such as (H_0: $\overline{v}_0 = \underline{v}_0$) *against* ($H_1$: $\overline{v}_0 \neq \underline{v}_0$).

a statistical evaluation and thus give a probabilistic guarantee that the gap is smaller than some ε, not an almost sure one.

A different approach consists in building polyhedral inner approximations $\overline{V}_t$ of the Bellman functions V_t at each stage t, that is, $\overline{V}_t \geq V_t$. A deterministic upper bound $\overline{V}_0(x_0)$ of the optimal value of problem (1.1) thus becomes available, and it is then possible to perform a stopping test of the SDDP algorithm on the almost-sure gap $\overline{V}_0(x_0) - \underline{V}_0(x_0)$. This path is followed in [17]. More precisely, starting from a polyhedral inner approximation $\overline{V}_{t+1}$ at time $t+1$ and choosing an arbitrary sequence of points $\{x_t^j\}_{j\in[\![1,J_t]\!]}$, the authors show how to compute a value q_t^j at each point x_t^j such that $q_t^j \geq V_t(x_t^j)$. The inner polyhedral approximation $\overline{V}_t$ is then obtained from the pairs $\{(x_t^j, q_t^j)\}_{j\in[\![1,J_t]\!]}$. A delicate issue when devising this inner approximation is the choice of the points x_t^j defining the polyhedral function $\overline{V}_t$. The authors suggest using points generated by some other algorithm, such as SDDP. They compute the upper bound only once the algorithm ends, and hence this upper bound is not used inside a stopping test. Moreover, they use the inner approximations $\overline{V}_t$ to devise a policy and show that the expected value of this policy is less than the upper bound $\overline{V}_0(x_0)$. Another approach involving inner and outer approximations of the Bellman functions is described in [1], where both a lower and an upper bound are updated at each stage and each iteration of the SDDP algorithm. Further, they use the difference between upper and lower bounds to select a specific noise trajectory (instead of a random noise selection, as is typical in SDDP). The convergence of the resulting fully deterministic algorithm is proved. A deterministic method involving inner and outer approximations of Bellman functions was also developed in [9] for the robust dynamic optimization framework. The main difference compared with SDDP is that [9] uses an inner approximation to determine a worst-case scenario for the forward pass, while computing a state trajectory and updating the lower approximation as usual. The inner approximation is updated directly from a primal perspective, which is equivalent to the dual update presented in this paper. Ultimately, let us mention the approach described in [5], using inner and outer approximations to alleviate the curse of dimensionality of dynamic programming by relaxing the demand for optimality so that the distance to the optimal solution is kept within prespecified bounds.

1.3. Contents of the paper. In section 2, we introduce the formalism of linear Bellman operators for a large class of stochastic optimization problems, we define its dual linear Bellman operator, and we enlighten the relationship between them thanks to Fenchel conjugacy. We also present an abstract SDDP algorithm that applies to a sequence of functions recursively defined through linear Bellman operators. In section 3, we use the conjugacy results obtained in section 2 to obtain a recursion on the dual value functions, on which we apply the abstract SDDP algorithm, yielding a dual SDDP algorithm for solving problem (1.1). The main result of this section is that we eventually obtain a converging exact upper bound over the value of problem (1.1). In section 4, we show how to build inner approximations of Bellman functions associated with problem (1.1) thanks to the outer approximations computed by the dual SDDP algorithm. These inner approximations induce a control strategy converging toward an optimal one (see Theorem 4.4). Furthermore, the expected cost incurred by this strategy is shown to be lower than the exact upper bound obtained in section 3. Ultimately, in section 5, we illustrate all the presented methodology on an energy management problem inspired by Électricité de France, at the European scale. The results show, on the one hand, that having at our disposal an exact upper bound in SDDP allows us to devise a more efficient stopping test for SDDP than the usual ones based on a Monte Carlo approach and, on the other hand, that the strategy based

on the inner approximations of the Bellman functions outperforms the usual strategy obtained using standard outer approximations.

1.4. Notation.

- $[\![i,j]\!]$ denotes the set of integers between i and j.
- Ω denotes a finite set of cardinality $|\Omega|$ supporting a probability distribution $\mathbb{P}$: $\Omega = \{\omega_1, \dots, \omega_{|\Omega|}\}$ and $\mathbb{P}(\{\omega_i\}) = \pi_i$ for all $i \in [\![1, |\Omega|]\!]$.
- Random variables are denoted using bold uppercase letters (such as $\boldsymbol{Z} : \Omega \to \mathbb{Z}$), and their realizations are denoted using lowercase letters ($z \in \mathbb{Z}$).
- $\boldsymbol{X} : \Omega \to \mathbb{X}$ corresponds to the state, $\boldsymbol{U} : \Omega \to \mathbb{U}$ to the control, $\boldsymbol{\xi} : \Omega \to \Xi$ to the noise.
- $[R]^\star$ denotes the Fenchel transform of an extended real-valued function R.
- $V_t : \mathbb{X}_t \to \mathbb{R}$ is the Bellman value function associated to problem (1.1) at time t.
- $F_t = [V_t]^\star$ is the dual value function associated with problem (1.1) at time t.
- $\mathcal{B}$ is the Bellman operator associated with a generic linear problem, with associated solution operator $\mathcal{S}$, and dual operator denoted $\mathcal{B}^\ddagger$.
- $\mathcal{T}$ is the Bellman operator associated with problem (1.1), its dual being denoted $\mathcal{T}^\ddagger$.
- Underlined notation (e.g., $\underline{V}$) corresponds to a lower approximation of a function (e.g., V). Overlined notation (e.g., $\overline{V}$) denotes an upper approximation.
- $\mathbb{I}_{x \in X}$ is the indicator function with value 0 if $x \in X$ and $+\infty$ otherwise.
- $\mathrm{dom}(R) := \{x \in \mathbb{R}^n \mid R(x) < +\infty\}$ is the (effective) domain of $R : \mathbb{R}^n \to \bar{\mathbb{R}}$.

2. Linear Bellman operators. This self-contained section is devoted to the definition and properties of linear Bellman operators (LBOs). In subsection 2.1 we present the abstract formalism of LBOs that allows us to write our subsequent dynamic programming equations in a compact manner. In subsection 2.2 we show that the Fenchel transform of an LBO is also an LBO. In subsection 2.3 we present an abstract version of the SDDP algorithm adapted to the LBO formulation.

2.1. The formalism of linear Bellman operators. We first introduce the notion of *linear Bellman operator*, which is a particular class of Bellman operators (see, e.g., the mappings defined in [4, section 1.1]) associated with stochastic optimal control problems where costs and constraints are linear.

We consider an abstract probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Recall that Ω is a finite set (see Assumption 1.1) and assume that the σ-field $\mathcal{A}$ is generated by all the singletons of Ω. We denote by $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ the set of functions defined on $\mathbb{R}^{n_x}$ and taking values in $[-\infty, +\infty]$, and by $\mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})$ the space of $\mathbb{R}^{n_x}$-valued measurable functions defined on $(\Omega, \mathcal{A}, \mathbb{P})$.

DEFINITION 2.1. *An operator $\mathcal{B} : \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ is said to be a* linear Bellman operator *(LBO) if for all $R \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$, we have*

$$\mathcal{B}(R) : x \mapsto \inf_{(\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{G}(x)} \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big], \tag{2.1}$$

where $\mathcal{G} : \mathbb{R}^{n_x} \rightrightarrows \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u + n_x})$ is the set-valued mapping defined by

$$\mathcal{G}(x) := \big\{(\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u + n_x}) \quad | \quad Tx + \mathcal{W}_u(\boldsymbol{U}) + \mathcal{W}_y(\boldsymbol{Y}) \leq \boldsymbol{H}\big\}. \tag{2.2}$$

In (2.2), $\mathcal{W}_u : \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u}) \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c})$ and $\mathcal{W}_y : \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c})$ are two linear *operators. Here, $\boldsymbol{U}$ and $\boldsymbol{Y}$ are two decision random variables taking values on $\mathbb{R}^{n_u}$ and $\mathbb{R}^{n_x}$, respectively. The two random variables $\boldsymbol{C} : \Omega \to \mathbb{R}^{n_u}$*

and $\boldsymbol{H} : \Omega \to \mathbb{R}^{n_c}$ *are exogenous uncertainties in problem* (2.1)*, and we note that* $\boldsymbol{\xi} = (\boldsymbol{C}, \boldsymbol{H})$*. The deterministic matrix* $T \in \mathbb{R}^{n_c \times n_x}$ *is given data.*

We define the domain $\mathrm{dom}(\mathcal{G}) := \left\{x \in \mathbb{R}^{n_x} \mid \mathcal{G}(x) \neq \emptyset\right\}$ *and say that* $\mathcal{B}$ *is* compact *if* $\mathcal{G}$ *is compact-valued with nonempty compact domain.*

Finally, for any $R \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$*, we denote by* $\mathcal{S}(R)$ *the set-valued mapping giving, for any* $x \in \mathbb{R}^{n_x}$*, the set of optimal solutions* $\boldsymbol{Y}$ *of problem* (2.1)*, that is,* $\mathcal{S}(R) : \mathbb{R}^{n_x} \rightrightarrows \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})$*, with*

(2.3)
$$\mathcal{S}(R) : x \rightrightarrows \underset{\boldsymbol{Y} \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})}{\arg\min} \left(\inf_{\boldsymbol{U} \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u})} \left\{ \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big] \quad \middle| \quad (\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{G}(x) \right\} \right).$$

From this very definition, we deduce the following inclusion:

$$\mathrm{dom}\left(\mathcal{B}(R)\right) \subset \mathrm{dom}(\mathcal{G}) \,. \tag{2.4}$$

Example 2.2. We give some classical examples of operators $\mathcal{W}_u$ and $\mathcal{W}_y$ involved in Definition 2.1 of $\mathcal{B}$. We stress that $\mathcal{W} : \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c})$ is a linear operator over a space of random variables, and we describe the associated adjoint operator, that is, the linear operator $\mathcal{W}^\dagger$ such that $\left\langle \boldsymbol{X} \,, \mathcal{W}(\boldsymbol{Y}) \right\rangle = \left\langle \mathcal{W}^\dagger(\boldsymbol{X}) \,, \boldsymbol{Y} \right\rangle$, with $\left\langle \boldsymbol{X} \,, \boldsymbol{Y} \right\rangle = \mathbb{E}\big[\boldsymbol{X}^\top \boldsymbol{Y}\big]$.

- Linear pointwise operator: $\mathcal{W}\big(\omega \mapsto \boldsymbol{Y}(\omega)\big) = \big(\omega \mapsto A\boldsymbol{Y}(\omega)\big)$. Such an operator allows us to encode a *pointwise constraint*, and $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \boldsymbol{X}$.
- Linear expected operator: $\mathcal{W}\big(\omega \mapsto \boldsymbol{Y}(\omega)\big) = \big(\omega \mapsto A\mathbb{E}[\boldsymbol{Y}]\big)$. Such an operator allows us to encode a *constraint in expectation*, and $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \mathbb{E}[\boldsymbol{X}]$.
- Linear conditional operator: $\mathcal{W}\big(\omega \mapsto \boldsymbol{Y}(\omega)\big) = \big(\omega \mapsto A\mathbb{E}[\boldsymbol{Y} \,|\mathcal{F}](\omega)\big)$, where $\mathcal{F}$ is a sub-σ-field of $\mathcal{A}$. Such an operator allows us to encode, for example, a measurability constraint and $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \mathbb{E}[\boldsymbol{X} \,|\mathcal{F}]$.

Of course, any linear combination of these three kinds of operator is also linear.

We recall the key notion of *relatively complete recourse* introduced in [20].

Definition 2.3. *Let* $R \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ *and let* $\mathcal{B}$ *be an LBO. We say that the pair* $(\mathcal{B}, R)$ *satisfies the* relatively complete recourse *(RCR) assumption if*

$$\forall x \in \mathrm{dom}(\mathcal{G}) \,, \;\; \exists (\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{G}(x) \;\; \textit{such that} \;\; \boldsymbol{Y}(\omega) \in \mathrm{dom}(R) \quad \forall \omega \in \Omega \,. \tag{2.5}$$

Note that if the pair $(\mathcal{B}, R)$ satisfies the RCR assumption, then

$$\mathrm{dom}\left(\mathcal{B}(R)\right) = \mathrm{dom}(\mathcal{G}) \,. \tag{2.6}$$

If, in addition, $\mathcal{B}$ is compact, then $\mathcal{B}(R)$ is finite at some point x_0. We introduce the seemingly stronger notion of R-compatibility. These two notions are in fact equivalent (for polyhedral functions), as Lemma 2.5 shows that if $(\mathcal{B}, R)$ satisfies the RCR assumption, then there exists an equivalent R-compatible $\widetilde{\mathcal{B}}$ operator.

Definition 2.4. *Let* $R \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ *be a polyhedral function and* $\mathcal{B}$ *an LBO. We say that* $\mathcal{B}$ *is* R*-compatible if*

$$\forall x \in \mathrm{dom}(\mathcal{G}) \,, \;\; \forall (\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{G}(x) \,, \;\; \boldsymbol{Y}(\omega) \in \mathrm{dom}(R) \quad \forall \omega \in \Omega \,. \tag{2.7}$$

Lemma 2.5. *Consider* $R \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ *a polyhedral function and* $\mathcal{B}$ *an LBO. We define the LBO* $\widetilde{\mathcal{B}}$ *as follows: for all* $Q \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$*,*

$$\widetilde{\mathcal{B}}(Q) : x \mapsto \inf_{(\boldsymbol{U}, \boldsymbol{Y}) \in \widetilde{\mathcal{G}}(x)} \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + Q(\boldsymbol{Y})\Big] \,,$$

with $\widetilde{\mathcal{G}}(x) := \big\{(\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u+n_x}) \mid Tx + \mathcal{W}_u(\boldsymbol{U}) + \mathcal{W}_y(\boldsymbol{Y}) \leq \boldsymbol{H} \ \text{ and } \ \boldsymbol{Y} \in \mathrm{dom}(R)\big\}$.

If the pair $(\mathcal{B}, R)$ *satisfies an RCR assumption, then the LBO* $\widetilde{\mathcal{B}}$ *is* R*-compatible, with* $\mathrm{dom}\left(\mathcal{G}\right) = \mathrm{dom}\left(\widetilde{\mathcal{G}}\right)$*, and* $\mathcal{B}(R) = \widetilde{\mathcal{B}}(R)$.

Proof. By construction, $\widetilde{\mathcal{B}}$ is R-compatible and $\mathrm{dom}\left(\widetilde{\mathcal{G}}\right) \subset \mathrm{dom}\left(\mathcal{G}\right)$. Let $x \in \mathrm{dom}\left(\mathcal{G}\right)$. The RCR assumption implies that $x \in \mathrm{dom}\left(\widetilde{\mathcal{G}}\right)$. Moreover, the constraint $\boldsymbol{Y} \in \mathrm{dom}(R)$ is implicit in the definition of $\mathcal{B}(R)$ since $\boldsymbol{C}^\top \boldsymbol{U} + R(\boldsymbol{Y}) = +\infty$ if $\boldsymbol{Y} \notin \mathrm{dom}(R)$, hence the result. ∎

We now give some properties of LBOs and polyhedral functions, whose proofs are given in Appendix A. First, following [19, section 19], we recall some results about polyhedral functions. A convex extended real-valued function is *proper* if it never takes the value $-\infty$ and is not identically equal to $+\infty$. A *polyhedral subset* of $\mathbb{R}^n$ is a finite intersection of closed half spaces, and a *polyhedral function* is a function whose epigraph is a polyhedral set. In particular a polyhedral function is convex lower semicontinuous, but not necessarily proper. A nonproper polyhedral function takes the value $-\infty$ on a polyhedral set, and $+\infty$ elsewhere. The proofs of the next two propositions are given in Appendix A.

PROPOSITION 2.6. *Let* R *be a function of* $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$*, and let* $\mathcal{B}$ *be an LBO. Then we have the following properties:*

1. *If* R *is convex, then* $\mathcal{B}(R)$ *is convex.*
2. *If* R *is polyhedral, then* $\mathcal{B}(R)$ *is polyhedral.*
3. *If* $R \geq \tilde{R}$*, then* $\mathcal{B}(R) \geq \mathcal{B}(\tilde{R})$.

Remark 2.7. Assume that function R is proper and polyhedral. Then if $\mathcal{B}(R)$ is finite at some point, $\mathcal{B}(R)$ is a proper polyhedral function. Furthermore, if $\mathcal{B}(R)(x)$ is finite, solving its (linear programming) dual generates a supporting hyperplane of the function $\mathcal{B}(R)$ at point x, that is, a pair $(\lambda, \beta) \in \mathbb{R}^{n_x} \times \mathbb{R}$ such that $\langle \lambda, \cdot \rangle + \beta \leq \mathcal{B}(R)(\cdot)$ and $\langle \lambda\,, x \rangle + \beta = \mathcal{B}(R)(x)$. Such hyperplanes, or cuts, are of paramount importance for the SDDP algorithm.

The next proposition establishes a link between the Lipschitz constants of R and $\mathcal{B}(R)$. It can be proved similarly to Proposition 2.7 in [22].

PROPOSITION 2.8. *Let* R *be a proper polyhedral function of* $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$*, and let* $\mathcal{B}$ *be an LBO. Assume that* $(\mathcal{B}, R)$ *satisfies the RCR assumption and that* $\mathcal{B}(R)$ *is finite at some point. If* R *is Lipschitz (for the* L_1*-norm) with constant* L_R*, then* $\mathcal{B}(R)$ *is also Lipschitz on its domain (which is* $\mathrm{dom}(\mathcal{G})$*) with constant* $\phi(L_R) = (\|\boldsymbol{C}\|_\infty + L_R)\kappa_W \|T\|_\infty$*, where* κ_W *is a constant associated with the linear operator* $(\mathcal{W}_u, \mathcal{W}_y)$.

2.2. Fenchel transform of a linear Bellmen operator. Let f be an extended real-valued function. Its Fenchel conjugate $f^\star$ and its concave conjugate $f_\star$ are defined as

$$f^\star(\lambda) := \sup_{x \in \mathbb{R}^n} \langle \lambda\,, x \rangle - f(x) \qquad \text{and} \qquad f_\star(\lambda) := \inf_{x \in \mathbb{R}^n} \langle \lambda\,, x \rangle - f(x)\,. \tag{2.8}$$

We recall the following theorem (see [19, section 31]).

THEOREM 2.9 (Fenchel's duality theorem). *Let* f *and* $-g$ *be proper polyhedral functions defined on* $\mathbb{R}^n$*. If* $\mathrm{dom}(f) \cap \mathrm{dom}(-g) \neq \emptyset$*, then we have*

$$\inf_{x \in \mathbb{R}^n} f(x) - g(x) = \sup_{\lambda \in \mathbb{R}^n} g_\star(\lambda) - f^\star(\lambda)\,.$$

We now define the *dual linear Bellman operator* $\mathcal{B}^\ddagger$ of an LBO $\mathcal{B}$.

DEFINITION 2.10. *Let $\mathcal{B}$ be an LBO (see Definition 2.1). We denote by $\mathcal{B}^\ddagger$ the dual LBO of $\mathcal{B}$, defined for a given function $Q \in \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}})$ by*

$$\mathcal{B}^\ddagger(Q) : \lambda \mapsto \inf_{(\mu,\nu)\in\mathcal{G}^\ddagger(\lambda)} \mathbb{E}\Big[-\boldsymbol{\mu}^\top \boldsymbol{H} + Q(\boldsymbol{\nu})\Big], \tag{2.9}$$

where

$$\mathcal{G}^\ddagger(\lambda) = \big\{(\boldsymbol{\mu},\boldsymbol{\nu}) \in \mathcal{L}^0(\Omega,\mathcal{A};\mathbb{R}^{n_c+n_x}) \,|\, T^\top \mathbb{E}\big[\boldsymbol{\mu}\big] + \lambda = 0, \mathcal{W}_u^\dagger(\boldsymbol{\mu}) = \boldsymbol{C}, \mathcal{W}_y^\dagger(\boldsymbol{\mu}) = \boldsymbol{\nu}, \boldsymbol{\mu} \le 0\big\}, \tag{2.10}$$

$\mathcal{W}_u^\dagger$ (resp., $\mathcal{W}_y^\dagger$) being the adjoint operator of $\mathcal{W}_u$ (resp., $\mathcal{W}_y$).

Note that straightforward computations show that $(\mathcal{B}^\ddagger)^\ddagger = \mathcal{B}$.

Calling $\mathcal{B}^\ddagger$ the dual of $\mathcal{B}$ is justified by the following theorem.

THEOREM 2.11. *Let R be a proper polyhedral function, and let $\mathcal{B}$ be a compact LBO (see Definition 2.1), such that the pair $(\mathcal{B}, R)$ satisfies the RCR assumption. Then $\mathcal{B}(R)$ is a proper polyhedral function and its Fenchel transform is given by*

$$[\mathcal{B}(R)]^\star = \mathcal{B}^\ddagger\big([R]^\star\big)\,. \tag{2.11}$$

Proof. First note that as $\mathcal{B}$ is compact, $\mathcal{G}$ has a nonempty compact domain, and thus $\mathcal{B}(R)$ is finite at some point by the RCR assumption. We denote $\langle \boldsymbol{X}, \boldsymbol{Y}\rangle = \mathbb{E}\big[\boldsymbol{X}^\top \boldsymbol{Y}\big]$, and

$$\mathcal{R}(\boldsymbol{Y}) = \mathbb{E}\big[R(\boldsymbol{Y})\big], \quad \mathcal{K}(x,\boldsymbol{Y}) = \min_{\boldsymbol{U}} \ \big\{ \langle \boldsymbol{C}, \boldsymbol{U}\rangle \mid Tx + \mathcal{W}_u(\boldsymbol{U}) + \mathcal{W}_y(\boldsymbol{Y}) \le \boldsymbol{H}\big\}\,.$$

By definition, we have $\mathcal{B}(R)(x) = \inf_{\boldsymbol{Y}} \mathcal{K}(x,\boldsymbol{Y}) + \mathcal{R}(\boldsymbol{Y})$. Thus, for any $\lambda \in \mathbb{R}^{n_x}$, we have

$$\begin{aligned}[\mathcal{B}(R)]^\star(\lambda) &= \sup_{x\in\mathbb{R}^{n_x}} \Big\{ x^\top\lambda - \inf_{\boldsymbol{Y}} \big\{\mathcal{K}(x,\boldsymbol{Y}) + \mathcal{R}(\boldsymbol{Y})\big\}\Big\} \\ &= -\inf_{\boldsymbol{Y}} \Big\{ \mathcal{R}(\boldsymbol{Y}) - \sup_{x\in\mathbb{R}^{n_x}} x^\top\lambda - \mathcal{K}(x,\boldsymbol{Y})\Big\}\,.\end{aligned}$$

As R is a proper polyhedral function, so is $\mathcal{R}$. By construction, $\mathcal{K}$ is polyhedral. Since $\mathcal{B}$ is a compact LBO, the minimization in $\boldsymbol{U}$ is over a compact set, so that $\mathcal{K}$ is never equal to $-\infty$. Furthermore, $\mathcal{K}$ is proper as $\mathrm{dom}(\mathcal{B}(R)) = \mathrm{dom}(\mathcal{G}) \neq \emptyset$.

Let $\Phi(\boldsymbol{Y}) := \sup_{x\in\mathbb{R}^{n_x}} x^\top\lambda - \mathcal{K}(x,\boldsymbol{Y})$. Note that, for $x \notin \mathrm{dom}(\mathcal{G})$, we have $\mathcal{K}(x,\boldsymbol{Y}) = +\infty$, and thus $\Phi(\boldsymbol{Y}) = \sup_{x\in\mathrm{dom}(\mathcal{G})} x^\top\lambda - \mathcal{K}(x,Y)$. Consequently, as $\mathrm{dom}(\mathcal{G})$ is a compact set, we deduce that $-\Phi$ is a proper polyhedral function. Finally, the RCR assumption ensures that $\mathrm{dom}(-\Phi) \cap \mathrm{dom}(\mathcal{R}) \neq \emptyset$. Now, using Fenchel duality (see Theorem 2.9), we have that

$$[\mathcal{B}(R)]^\star(\lambda) = -\sup_{\boldsymbol{\nu}} \Phi_\star(\boldsymbol{\nu}) - \mathcal{R}^\star(\boldsymbol{\nu}) = \inf_{\boldsymbol{\nu}} \mathcal{R}^\star(\boldsymbol{\nu}) - \Phi_\star(\boldsymbol{\nu})\,,$$

where $\mathcal{R}^\star(\boldsymbol{\nu}) = \mathbb{E}\big[R^\star(\boldsymbol{\nu})\big]$ and

$$\begin{aligned}
\Phi_\star(\boldsymbol{\nu}) &= \inf_{\boldsymbol{Y}} \big\langle \boldsymbol{\nu}\,, \boldsymbol{Y} \big\rangle - \Phi(\boldsymbol{Y}) \\
&= \inf_{\boldsymbol{Y}} \big\langle \boldsymbol{\nu}\,, \boldsymbol{Y} \big\rangle - \sup_{x} \big\{ x^\top \lambda - \mathcal{K}(x, \boldsymbol{Y}) \big\} \\
&= \inf_{x, \boldsymbol{Y}, \boldsymbol{U}} \big\{ \big\langle \boldsymbol{\nu}\,, \boldsymbol{Y} \big\rangle - x^\top \lambda + \big\langle \boldsymbol{C}\,, \boldsymbol{U} \big\rangle \;\big|\; Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \big\} \\
&= \inf_{x, \boldsymbol{Y}, \boldsymbol{U}} \big\langle \boldsymbol{\nu}\,, \boldsymbol{Y} \big\rangle - x^\top \lambda + \big\langle \boldsymbol{C}\,, \boldsymbol{U} \big\rangle + \sup_{\boldsymbol{\mu} \leq 0} \big\langle -\boldsymbol{\mu}\,, Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) - \boldsymbol{H} \big\rangle .
\end{aligned}$$

As $\mathrm{dom}(\mathcal{G})$ is nonempty and compact, there exists a primal feasible solution to the above linear program, and by duality we have

$$\begin{aligned}
\Phi_\star(\boldsymbol{\nu}) &= \sup_{\boldsymbol{\mu} \leq 0} \big\langle \boldsymbol{\mu}\,, \boldsymbol{H} \big\rangle + \inf_{x} \big\{ - x^\top \lambda - \big\langle T^\top \boldsymbol{\mu}\,, x \big\rangle \big\} + \inf_{\boldsymbol{Y}} \big\{ \big\langle \boldsymbol{\nu}\,, \boldsymbol{Y} \big\rangle - \big\langle \mathcal{W}_y^\dagger(\boldsymbol{\mu})\,, \boldsymbol{Y} \big\rangle \big\} \\
&\qquad + \inf_{\boldsymbol{U}} \big\{ \big\langle \boldsymbol{C}\,, \boldsymbol{U} \big\rangle - \big\langle \mathcal{W}_u^\dagger(\boldsymbol{\mu})\,, \boldsymbol{U} \big\rangle \big\} \\
&= \sup_{\boldsymbol{\mu} \leq 0} \big\{ \big\langle \boldsymbol{\mu}\,, \boldsymbol{H} \big\rangle \;|\; T^\top \mathbb{E}(\boldsymbol{\mu}) + \lambda = 0\,,\;\; \mathcal{W}_u^\dagger(\boldsymbol{\mu}) = \boldsymbol{C}\,,\;\; \mathcal{W}_y^\dagger(\boldsymbol{\mu}) = \boldsymbol{\nu} \big\}\,.
\end{aligned}$$

Finally,

$$[\mathcal{B}(R)]^\star(\lambda) = \inf_{\boldsymbol{\nu}, \boldsymbol{\mu} \leq 0} \Big\{ \mathbb{E}\big[-\boldsymbol{\mu}^\top \boldsymbol{H} + R^\star(\boldsymbol{\nu}) \big] \,\Big|\, T^\top \mathbb{E}\big[\boldsymbol{\mu}\big] + \lambda = 0\,,\;\; \mathcal{W}_u^\dagger(\boldsymbol{\mu}) = \boldsymbol{C}\,,\;\; \mathcal{W}_y^\dagger(\boldsymbol{\mu}) = \boldsymbol{\nu} \Big\},$$

which ends the proof. ∎

2.3. An abstract SDDP algorithm. We now consider a sequence of functions $\big\{R_t\big\}_{t \in [\![0,T]\!]}$ that follows the Bellman backward recursion

$$\begin{cases} R_T = K, \\ R_t = \mathcal{B}_t(R_{t+1}) \quad \forall t \in [\![0, T-1]\!]\,, \end{cases} \tag{2.12}$$

where K is a proper polyhedral function, and where $\big\{\mathcal{B}_t\big\}_{t \in [\![0,T-1]\!]}$ is a sequence of LBOs defined as in Definition 2.1, with the associated random variables $(\boldsymbol{C}_{t+1}, \boldsymbol{H}_{t+1})$, linear operators $(\mathcal{W}^u_{t+1}, \mathcal{W}^y_{t+1})$, and mappings $(\mathcal{G}_t, \mathcal{S}_t)$ indexed by t.

The notion of R-compatibility given in Definition 2.4 is extended to the dynamic case as follows.

DEFINITION 2.12. *Let* $\big\{\mathcal{B}_t\big\}_{t \in [\![0,T-1]\!]}$ *be a sequence of LBOs such that* $\mathrm{dom}(\mathcal{G}_t) \neq \emptyset$ *for all* $t \in [\![0, T-1]\!]$, *and let* $\big\{R_t\big\}_{t \in [\![0,T]\!]}$ *be defined by the Bellman recursion* (2.12). *We say that the sequence* $\big\{\mathcal{B}_t\big\}_{t \in [\![0,T-1]\!]}$ *is* K-compatible *if*

$$\begin{aligned} &\forall t \in [\![0, T-1]\!], \quad \forall x \in \mathrm{dom}(\mathcal{G}_t), \quad \forall (\boldsymbol{U}_{t+1}, \boldsymbol{Y}_{t+1}) \in \mathcal{G}_t(x), \\ &\boldsymbol{Y}_{t+1}(\omega) \in \mathrm{dom}(\mathcal{G}_{t+1}) \qquad \forall \omega \in \Omega\,, \end{aligned} \tag{2.13}$$

where by convention $\mathrm{dom}(\mathcal{G}_T) = \mathrm{dom}(K)$.

A direct consequence of this definition is that, for all $t \in [\![0, T]\!]$, $\mathrm{dom}(R_t) = \mathrm{dom}(\mathcal{G}_t)$.

Remark 2.13. A natural extension of the RCR assumption (2.5) to a sequence of pairs $\{(\mathcal{B}_t, R_{t+1})\}_{t\in[\![0,T-1]\!]}$ would be asking that, for all $t \in [\![0, T-1]\!]$, the pair $(\mathcal{B}_t, R_{t+1})$ satisfies the RCR assumption, that is,

$$\begin{aligned} &\forall t \in [\![0,T-1]\!], \quad \forall x \in \mathrm{dom}(\mathcal{G}_t), \quad \exists (\boldsymbol{U}_{t+1}, \boldsymbol{Y}_{t+1}) \in \mathcal{G}_t(x), \\ &\boldsymbol{Y}_{t+1}(\omega) \in \mathrm{dom}(\mathcal{G}_{t+1}) \qquad \forall \omega \in \Omega\ . \end{aligned} \tag{2.14}$$

As seen in the static case, this seemingly weaker assumption is in fact equivalent to Definition 2.12. Without loss of generality, we will assume K-compatibility instead of (2.14) by adding in the definition of the LBO $\mathcal{B}_t$ the (implicit) constraint $\boldsymbol{Y}_{t+1} \in \mathrm{dom}(R_{t+1})$ $\mathbb{P}$-a.s. This last assumption proves useful in ensuring that all states generated in the forward pass of the SDDP algorithm are admissible.

SDDP is an algorithm that iteratively constructs finite lower polyhedral approximations of the sequence of functions $\left\{R_t\right\}_{t\in[\![0,T-1]\!]}$ given by (2.12). Starting from an initial point $x_0 \in \mathbb{R}^{n_x}$, the algorithm determines in a forward pass a sequence of states $\{x_t^k\}_{t\in[\![0,T]\!]}$ at which the approximations of the sequence $\left\{R_t\right\}_{t\in[\![0,T-1]\!]}$ will be refined in the backward pass. More precisely, a pseudocode describing the abstract SDDP algorithm is given in Algorithm 2.1. In the initialization of the algorithm, setting $\underline{R}_t^0$ to $-\infty$ is shorthand for saying that the first forward pass is arbitrary.

Algorithm 2.1 Abstract SDDP algorithm.

Data: Initial point x_0
$\underline{R}_t^0 \leftarrow -\infty$
for $k : 0,\ 1, \ldots$ **do**
  // Forward Pass : compute a set of trial points $\left\{x_t^k\right\}_{t\in[\![0,T]\!]}$
  $x_0^k \leftarrow x_0$
  for $t : 0$ **to** $T-1$ **do**
    select $\boldsymbol{X}_{t+1}^k \in \mathcal{S}_t(\underline{R}_{t+1}^k)(x_t^k)$ // see Definition 2.1
    Randomly select $\omega_t^k \in \Omega$
    $x_{t+1}^k \leftarrow \boldsymbol{X}_{t+1}^k(\omega_t^k)$
  end for
  // Backward Pass : refine the lower approximations at the trial points
  $\underline{R}_T^{k+1} \leftarrow K$
  for $t : T-1$ **to** 0 **do**
    $\theta_t^{k+1} \leftarrow \mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$; select $\lambda_t^{k+1} \in \partial\Big[\mathcal{B}_t(\underline{R}_{t+1}^{k+1})\Big](x_t^k)$ // see Remark 2.7
    $\beta_t^{k+1} \leftarrow \theta_t^{k+1} - \left\langle \lambda_t^{k+1}, x_t^k \right\rangle$
    $\underline{R}_t^{k+1} \leftarrow \max\left\{\underline{R}_t^k, \left\langle \lambda_t^{k+1}, \cdot \right\rangle + \beta_t^{k+1}\right\}$ // update lower approximation
  end for
  STOP if some stopping test is satisfied
end for

LEMMA 2.14. *Assume that $R^0(x_0)$ is finite and that $\left\{\mathcal{B}_t\right\}_{t\in[\![0,T-1]\!]}$ is a K-compatible sequence of LBOs. Then the abstract SDDP Algorithm* 2.1 *is well defined and there exists a real sequence $\left\{L_t\right\}_{t\in[\![0,T-1]\!]}$ such that each R_t given by* (2.12) *is L_t-Lipschitz on its domain. Moreover, for all $t \in [\![0, T-1]\!]$ and all $k \in \mathbb{N}$, the $\lambda_t^{(k)}$ produced by the algorithm satisfies $\|\lambda_t^{(k)}\|_\infty \leq L_t$.*

From Lemma 2.14, proven in Appendix B, we have the boundedness of $\lambda_t^{(k)}$, from

which we can easily adapt the proof of [10] in order to obtain the following convergence result.

PROPOSITION 2.15. *Assume that $R_0(x_0)$ is finite and that $\{\mathcal{B}_t\}_{t\in[\![0,T-1]\!]}$ is a K-compatible sequence of LBOs. Assume furthermore that, for each $t \in [\![0,T]\!]$, there exists a compact set X_t such that $x_t^k \in X_t$ for all k (this will be the case if $\{\mathcal{B}_t\}_{t\in[\![0,T-1]\!]}$ is a sequence of compact LBOs).*

Then the abstract SDDP Algorithm 2.1 *generates a nondecreasing sequence $\{\underline{R}_t^k\}_{k\in\mathbb{N}}$ of lower approximations of R_t, such that $\lim_{k\to+\infty}\underline{R}_0^k(x_0) = R_0(x_0)$.*

This algorithm is abstract in the sense that it only requires a sequence of LBOs. In the following section, we show how it can be applied to approximate the Bellman value functions $\{V_t\}_{t\in[\![0,T]\!]}$, or to approximate the Fenchel transforms of these Bellman functions.

3. Primal and dual SDDP.

In this section we recall the usual SDDP algorithm applied to problem (1.1). Next, leveraging the results of section 2, we introduce a dual SDDP algorithm, which is the abstract SDDP algorithm applied to the dual value functions. This eventually gives an exact upper bound of the value of problem (1.1). We denote by V_t the primal value functions, and by $F_t = [V_t]^\star$ the dual value functions obtained by the Fenchel transform.

3.1. Primal SDDP.

We first recall the standard (primal) SDDP algorithm.

3.1.1. Primal dynamic programming equations.

Thanks to the discrete white noise Assumption 1.1, we can solve problem (1.1) through dynamic programming, computing backward the value functions $\{V_t\}_{t\in[\![0,T]\!]}$ given by

$$\begin{cases} V_T = K\ , \\ V_t = \mathcal{T}_t\big(V_{t+1}\big)\ , \end{cases} \tag{3.1}$$

where the primal Bellman operator $\mathcal{T}_t : \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}})$ is defined as follows:

$$\mathcal{T}_t(R) : x \mapsto \inf_{\boldsymbol{U}_{t+1},\boldsymbol{X}_{t+1}} \quad \mathbb{E}\Big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}_{t+1} + R(\boldsymbol{X}_{t+1})\Big] \tag{3.2a}$$

$$\text{s.t.} \quad \boldsymbol{X}_{t+1} = A_t x + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1}\ , \tag{3.2b}$$

$$D_t x + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \le 0\ , \tag{3.2c}$$

$$\underline{u}_{t+1} \le \boldsymbol{U}_{t+1} \le \overline{u}_{t+1}\ ,\ \ \underline{x}_t \le x \le \overline{x}_t\ , \tag{3.2d}$$

$$\boldsymbol{X}_{t+1} \in X_{t+1}\ , \tag{3.2e}$$

where

$$X_t = \big\{x \in \mathbb{R}^{n_x} \mid \exists \boldsymbol{U}_{t+1}, \text{ such that } (x,\boldsymbol{U}_{t+1}) \text{ satisfy (3.2c)–(3.2d)}\big\}\ , \tag{3.3a}$$

$$X_T = \mathrm{dom}(K)\ . \tag{3.3b}$$

We assume that the final cost function K has a compact domain, and thus all X_t are compact. Constraint (3.2d) ensures that if x does not satisfy $\underline{x}_t \le x \le \overline{x}_t$, then $\mathcal{T}_t(R)(x) = +\infty$. Constraints (3.2d) and (3.2e) ensure that the operator $\mathcal{T}_t$ is a compact LBO (see Definition 2.1). The associated set-valued mapping $\mathcal{G}_t$ is defined by

$$\mathcal{G}_t(x) := \big\{(\boldsymbol{U}_{t+1},\boldsymbol{X}_{t+1}) \quad \text{satisfying (3.2b)–(3.2e)}\big\}\ .$$

For notational simplicity, we assume in the following that constraint (3.2e) is induced by constraint (3.2c). This is always possible if one introduces $\boldsymbol{U}_{t+2}$ and $\boldsymbol{\xi}_{t+2}$ in problem (3.2) and then replaces (3.2e) by, for all $\xi_{t+2} \in \mathrm{supp}(\boldsymbol{\xi}_{t+2})$,

$$D_{t+1}X_{t+1}^{\xi_{t+2}} + E_{t+2}U_{t+2}^{\xi_{t+2}} + G_{t+2}\xi_{t+2} \leq 0 \ ,$$
$$\underline{u}_{t+2} \leq U_{t+2}^{\xi_{t+2}} \leq \overline{u}_{t+2}, \quad \underline{x}_{t+1} \leq X_{t+1}^{\xi_{t+2}} \leq \overline{x}_{t+1} \ .$$

Therefore, by introducing the new variables U_{t+2} and X_{t+2} as shown above, we can always induce constraint (3.2e) by constraint (3.2c). Note that in most practical cases we can provide a simpler description of X_t than (3.3).

We make the following assumptions.

Assumption 3.1.

1. The function K is polyhedral with compact domain such that for all $x \in \mathrm{dom}(K)$ we have $\underline{x}_T \leq x \leq \overline{x}_T$.
2. The sequence of LBOs $\left\{\mathcal{T}_t\right\}_{t\in[\![0,T]\!]}$ is K-compatible.
3. Problem (1.1) is finite-valued.

The next lemma, whose proof can be found in Appendix B, gives the properties of the sequence of LBOs $\left\{\mathcal{T}_t\right\}_{t\in[\![0,T-1]\!]}$.

LEMMA 3.2. *Under Assumption* 3.1*, for any* $t \in [\![0, T-1]\!]$*, we have*

$$\mathcal{T}_t(R) : x \mapsto \mathbb{E}\left[\widehat{\mathcal{T}}_t(R)(x, \boldsymbol{\xi}_{t+1})\right] \ , \tag{3.4}$$

where

$$\widehat{\mathcal{T}}_t(R) : (x,\xi) \mapsto \inf_{u_{t+1}, x_{t+1}} \quad a_t^\top x + b_{t+1}^\top u_{t+1} + R(x_{t+1}) \tag{3.5a}$$
$$s.t. \quad x_{t+1} = A_t x + B_{t+1}u_{t+1} + C_{t+1}\xi \ , \tag{3.5b}$$
$$D_t x + E_{t+1}u_{t+1} + G_{t+1}\xi \leq 0 \ , \tag{3.5c}$$
$$\underline{u}_{t+1} \leq u_{t+1} \leq \overline{u}_{t+1} \ , \ \underline{x}_t \leq x \leq \overline{x}_t \ . \tag{3.5d}$$

To recover the optimal state trajectories from Bellman functions, we introduce for each t the set-valued mapping associated with $\widehat{\mathcal{T}}_t(R)$:

$$\widehat{\mathcal{S}}_t(R) : (x,\xi) \rightrightarrows \operatorname*{arg\,min}_{x_{t+1}} \left(\inf_{u_{t+1}} \quad a_t^\top x + b_{t+1}^\top u_{t+1} + R(x_{t+1}) \right) \tag{3.6a}$$
$$\text{s.t.} \quad x_{t+1} = A_t x + B_{t+1}u_{t+1} + C_{t+1}\xi \ , \tag{3.6b}$$
$$D_t x + E_{t+1}u_{t+1} + G_{t+1}\xi \leq 0 \ , \tag{3.6c}$$
$$\underline{u}_{t+1} \leq u_{t+1} \leq \overline{u}_{t+1} \ , \ \underline{x}_t \leq x \leq \overline{x}_t \ . \tag{3.6d}$$

3.1.2. Primal SDDP algorithm. We now apply the abstract SDDP algorithm presented in subsection 2.3 to the primal Bellman operator given by (3.2). We denote $\pi_t^\xi := \mathbb{P}(\boldsymbol{\xi}_t = \xi)$ for all $t \in [\![1, T]\!]$ and for all $\xi \in \mathrm{supp}(\boldsymbol{\xi}_t)$. The associated pseudocode is given in Algorithm 3.1.

Remark 3.3. Note that the primal Bellman operator (3.4) is a specialized version of the abstract Bellman operator used in (2.12), which only involves a pointwise operator in the constraints. Hence, in the forward pass we just have to compute $\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^k)(x_t^k, \xi_{t+1}^k)$ and do not need to compute $\mathcal{T}_t(\underline{V}_{t+1}^k)(x_t^k)$, which would involve a larger linear problem. Similarly, in the backward pass at time t, we have to solve a

Algorithm 3.1 Primal SDDP algorithm.

Data: initial point x_0, initial lower bounds $\underline{V}_t^0$ on V_t
$\underline{V}_t^0 \leftarrow -\infty$
for $k : 0,\ 1, \dots$ **do**
  // Forward Pass : compute a set of trial points $\left\{x_t^k\right\}_{t\in[\![0,T]\!]}$
  Draw a noise scenario $\left\{\xi_t^k\right\}_{t\in[\![1,T]\!]}$
  $x_0^k \leftarrow x_0$
  for $t : 0$ **to** $T-1$ **do**
    select $x_{t+1}^k \in \widehat{\mathcal{S}}_t(\underline{V}_{t+1}^k)(x_t^k, \xi_{t+1}^k)$ // see (3.6)
  end for
  // Backward Pass : refine the lower-approximations at the trial points
  $\underline{V}_T^{k+1} \leftarrow K$
  for $t : T-1$ **to** 0 **do**
    for $\xi \in \mathrm{supp}(\boldsymbol{\xi}_{t+1})$ **do**
      $\underline{\theta}_t^{\xi,k+1} \leftarrow \widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi)$; select $\underline{\lambda}_t^{\xi,k+1} \in \partial\Big[\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})\Big](x_t^k, \xi)$
    end for
    $\underline{\lambda}_t^{k+1} \leftarrow \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_{t+1}^{\xi}\underline{\lambda}_t^{\xi,k+1}$ // taking expectation
    $\underline{\beta}_t^{k+1} \leftarrow \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_{t+1}^{\xi}\Big(\underline{\theta}_t^{\xi,k+1} - \big\langle \underline{\lambda}_t^{\xi,k+1}, x_t^k \big\rangle\Big)$
    $\underline{V}_t^{k+1} \leftarrow \max\left\{\underline{V}_t^k\ ,\ \langle \underline{\lambda}_t^k\ , \cdot\rangle + \underline{\beta}_t^{k+1}\right\}$ // update lower approximation
  end for
  STOP if some stopping test is satisfied
end for

$|\mathrm{supp}(\boldsymbol{\xi}_{t+1})|$ linear problem of the form $\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi_{t+1}^s)$ instead of the larger linear problem $\mathcal{T}_t(\underline{V}_{t+1}^{k+1})(x_t^k)$, and then perform an expectation. We will show in what follows that this is no longer the case in the dual SDDP algorithm.

The following proposition has been known since [18].

PROPOSITION 3.4. *Under Assumptions* 1.1 *and* 3.1*, the primal SDDP algorithm yields a converging lower bound for the value of problem* (1.1)*, i.e., for all* $k \in \mathbb{N}$*,* $\underline{V}_0^k(x_0) \leq V_0(x_0)$ *and* $\lim_{k\to\infty} \underline{V}_0^k(x_0) = V_0(x_0)$*. Further, the strategy induced by* $\underline{V}_t^k$ *is converging toward an optimal strategy. More precisely, using OA for "outer approximation," we consider* $\boldsymbol{X}_{t+1}^{OA,k} \in \mathcal{S}_t\big(\underline{V}_{t+1}^k\big)\big(\boldsymbol{X}_t^{OA,k}\big)$ *and denote by* $C_0^{OA,k}(x_0)$ *the expected cost of this strategy. Then we have* $\lim_{k\to+\infty} C_0^{OA,k}(x_0) = V_0(x_0)$*.*

Proof. By Assumption 1.1 the sequence of value functions $\{V_t\}_{t\in[\![0,T]\!]}$ can be obtained by dynamic programming and follows the backward recursion (3.1). By Assumption 3.1, we have the K-compatibility of the sequence of LBOs $\{\mathcal{T}_t\}_{t\in[\![0,T-1]\!]}$. Furthermore, as $\mathcal{T}_t$ is a compact LBO for any $t \in [\![0, T-1]\!]$, the sequence $\{x_t^k\}_{k\in\mathbb{N}}$ generated by the algorithm remains in a compact set. Hence, we can apply Proposition 2.15. The convergence proof of the strategy induced by $\underline{V}_t^{(k)}$ can be found in [10] or [18]. ∎

3.2. Dual SDDP. We present here a dual SDDP algorithm, which leverages the conjugacy results of subsection 2.2. We show that the Fenchel conjugates of the

primal value functions $\{V_t\}_{t\in[\![0,T]\!]}$ follow a recursive equation on which we apply the abstract SDDP Algorithm 2.1.

3.2.1. Lipschitz regularization. Recall that, for any proper functions f and g of $\mathbb{R}^n$, the infimal convolution of f and g is defined as $f\square g : x \mapsto \inf_{y\in\mathbb{R}^n} f(y)+g(x-y)$. We have the following result (see [2, Chapter 12]).

PROPOSITION 3.5. *Let f be a proper function of $\mathbb{R}^n$, and let L be a positive number. Then $f^L := f\square(L\|\cdot\|_1)$ is the largest L-Lipschitz function that is lower than f. The function f^L is called the L-Lipschitz regularization[4] of f.*

3.2.2. Dual dynamic programming equations. Using Definition 2.10, we can compute an explicit formulation of $\mathcal{T}_t^\ddagger$. Unfortunately the dual operators $\mathcal{T}_t^\ddagger$ are not necessarily compatible. Thus, we construct another sequence of LBO by, for all Q,

$$\mathcal{T}^\ddagger_{t,L_{t+1}}\Big(Q\Big) = \mathcal{T}_t^\ddagger\Big(Q + \mathbb{I}_{B_\infty(0,L_{t+1})}\Big), \tag{3.7}$$

where $B_\infty(0,L_{t+1})$ is the L_∞-ball of radius L_{t+1} centered in 0. We show that these operators are compatible and can be used in the dynamic recursion.

Now, consider the primal Bellman operator $\mathcal{T}_t : \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}})$ defined by (3.2). Ignoring the constant term $a_t^\top x$ and assuming that the last constraint (added to allow the K-compatibility of $\mathcal{T}_t$) has been embedded in constraint (3.2c), we can rewrite $\mathcal{T}_t^\ddagger$ as an LBO (see Definition 2.1) with the notation

$$\begin{aligned}
T_t &= [A_t \;\; -A_t \;\; 0 \;\; 0 \;\; -I \;\; I \;\; D_t]^\top ,\\
\mathcal{W}^u_{t+1}(\boldsymbol{U}) &= [B_{t+1} \;\; - \;\; B_{t+1} \;\; - \;\; I \;\; I \;\; 0 \;\; 0 \;\; E_{t+1}]^\top \boldsymbol{U} ,\\
\mathcal{W}^y_{t+1}(\boldsymbol{X}) &= [-I \;\; I \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^\top \boldsymbol{X} ,\\
\boldsymbol{H}_{t+1} &= [-C_{t+1}\boldsymbol{\xi}_{t+1} \;\; C_{t+1}\boldsymbol{\xi}_{t+1} \;\; -\underline{u}_{t+1} \;\; \overline{u}_{t+1} \;\; -\underline{x}_{t+1} \;\; \overline{x}_{t+1} \;\; - \;\; G_{t+1}\boldsymbol{\xi}_{t+1}]^\top .
\end{aligned}$$

Definition 2.10 allows us to compute the dual LBO $\mathcal{T}^\ddagger_{t,L_{t+1}}$ (see (3.7)). Denoting by $\boldsymbol{\mu} = (\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_7)$ the multiplier associated to the constraints of the primal problem (3.2), we obtain

$$\begin{aligned}
\mathcal{T}^\ddagger_{t,L_{t+1}}(Q)(\lambda) = \inf_{\boldsymbol{\mu},\boldsymbol{\nu}} \quad & \mathbb{E}\Big[-\boldsymbol{\mu}^\top \boldsymbol{H}_{t+1} + Q(\boldsymbol{\nu})\Big]\\
\text{s.t.} \quad & A_t^\top \mathbb{E}\big[\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big] - \mathbb{E}\big[\boldsymbol{\mu}_5 - \boldsymbol{\mu}_6\big] + D_t^\top \mathbb{E}\big[\boldsymbol{\mu}_7\big] + \lambda = 0 ,\\
& B_{t+1}^\top\big(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big) - \big(\boldsymbol{\mu}_3 - \boldsymbol{\mu}_4\big) + E_{t+1}^\top \boldsymbol{\mu}_7 - b_{t+1} = 0 ,\\
& -\big(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big) = \boldsymbol{\nu} ,\\
& \boldsymbol{\nu} \in B_\infty(0, L_{t+1}) ,\\
& \boldsymbol{\mu} \le 0 .
\end{aligned}$$

The following proposition gives the compatibility property of the dual LBOs.

PROPOSITION 3.6. *Under Assumption* 3.1*, the sequence $\big\{\mathcal{T}^\ddagger_{t,L_{t+1}}\big\}_{t\in[\![0,T-1]\!]}$ of LBOs defined by* (3.7) *is $[K]^\star$-compatible.*

Proof. Let λ be any element of $\mathbb{R}^n$, let $\boldsymbol{\mu}_7$ be an arbitrary nonpositive random variable, and let $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ be nonpositive random variables such that $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \in B_\infty(0, L_{t+1})$.

[4] Also called the Pasch–Hausdorff envelope.

We define[5]

$$\boldsymbol{\mu}_3 = \Big(B_{t+1}^\top(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + E_{t+1}^\top \boldsymbol{\mu}_7 - b_{t+1}\Big)^-,\ \boldsymbol{\mu}_4 = -\Big(B_{t+1}^\top(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + E_{t+1}^\top \boldsymbol{\mu}_7 - b_{t+1}\Big)^+,$$

$$\boldsymbol{\mu}_5 = \Big(A_t^\top \mathbb{E}\big[\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big] + D_t^\top \mathbb{E}\big[\boldsymbol{\mu}_7\big] + \lambda\Big)^-,\ \boldsymbol{\mu}_6 = -\Big(A_t^\top \mathbb{E}\big[\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big] + D_t^\top \mathbb{E}\big[\boldsymbol{\mu}_7\big] + \lambda\Big)^+,$$

$$\boldsymbol{\nu} = \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\ .$$

Then $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_7)$ is a nonpositive random variable by construction, and the pair $\big(\boldsymbol{\mu}, \boldsymbol{\nu}\big)$ satisfies all the constraints in the definition of $\mathcal{T}^\ddagger_{t,L_{t+1}}\big(Q\big)(\lambda)$. Such a pair $(\boldsymbol{\mu}, \boldsymbol{\nu})$ exists for all possible values of λ. Moreover, $\boldsymbol{\nu} \in B_\infty(0, L_{t+1}) \subset \mathbb{R}^n$. We thus deduce that the domain of the dual constraint set-valued mapping $\mathcal{G}^\ddagger_{t,L_{t+1}}$ (defined by (2.10)) is equal to the whole space $\mathbb{R}^n$, so that the sequence of dual linear Bellman operators $\big(\mathcal{T}^\ddagger_{t,L_{t+1}}\big)_{t\in[\![0,T-1]\!]}$ is compatible as $\mathrm{dom}\,\big([K]^\star\big) = \mathbb{R}^n$. ∎

THEOREM 3.7. *We assume that Assumption* 3.1 *holds true. For any $t \in [\![0,T]\!]$, we denote $F_t := [V_t]^\star$, where V_t is the Bellman value function obtained by* (3.1)*. For all $t \in [\![0,T]\!]$, let $L_t > 0$ be such that V_t is L_t-Lipschitz (for the L^1-norm) on its domain. Then the sequence of dual value functions $\big\{F_t\big\}_{t\in[\![0,T]\!]}$ satisfies the following backward recursion:*

$$F_T = [K]^\star\ , \tag{3.8a}$$

$$F_t = \mathcal{T}^\ddagger_{t,L_{t+1}}(F_{t+1}) \qquad \forall t \in [\![0,T-1]\!]\ . \tag{3.8b}$$

Proof. By Assumption 3.1, we have that $\big\{\mathcal{T}_t\big\}_{t=[\![0,T-1]\!]}$ is a K-compatible sequence of compact LBOs, with the associated sequence $\big\{V_t\big\}_{t=[\![0,T]\!]}$ of Bellman functions defined by (3.1). Let $t \in [\![0,T-1]\!]$. As V_{t+1} is L_{t+1}-Lipschitz continuous on $\mathrm{dom}(V_{t+1})$, V_{t+1} and $V_{t+1}^{L_{t+1}}$ coincide on $\mathrm{dom}(V_{t+1})$, and $\mathrm{dom}(V_{t+1}) = \mathrm{dom}(\mathcal{G}_{t+1})$ by the K-compatibility property (see Definition 2.12). We thus deduce that $\mathcal{T}_t\big(V_{t+1}^{L_{t+1}}\big) = \mathcal{T}_t\big(V_{t+1}\big) = V_t$. Theorem 2.11 applies, so that $[V_t]^\star = \mathcal{T}_t^\ddagger\big(\big[V_{t+1}^{L_{t+1}}\big]^\star\big)$. As V_{t+1} and $L_{t+1}\|\cdot\|_1$ take values in $(-\infty, +\infty]$, we have $\big[V_{t+1}^{L_{t+1}}\big]^\star = [V_{t+1}]^\star + \mathbb{I}_{B_\infty(0,L_{t+1})}$ (see [2, Corollary 13.24]). Thus we obtain $F_t = \mathcal{T}_t^\ddagger\big(F_{t+1} + \mathbb{I}_{B_\infty(0,L_{t+1})}\big)$. ∎

Remark 3.8. Lemma 2.14 shows how to find such a sequence of Lipschitz constants $\big\{L_t\big\}_{t\in[\![0,T]\!]}$. But in some cases we can directly derive a Lipschitz constant on the value functions and plug it into (3.8).

3.2.3. Dual SDDP algorithm. By Theorem 3.7, the dual value functions $\big\{F_t\big\}_{t\in[\![0,T]\!]}$ are solutions of a Bellman recursion involving linear Bellman operators $\big\{\mathcal{T}^\ddagger_{t,L_{t+1}}\big\}_{t\in[\![0,T-1]\!]}$, thus opening the door to the computation of outer approximations $\big\{\underline{F}_t^k\big\}_{t\in[\![0,T]\!]}$ of $\big\{F_t\big\}_{t\in[\![0,T]\!]}$ by SDDP (see Algorithm 3.2 below).

LEMMA 3.9. *For all $t \in [\![0,T-1]\!]$, $\big\{\big[\underline{F}_t^k\big]^\star\big\}_{k\in\mathbb{N}}$ is a decreasing sequence of upper approximations of the primal value function V_t: $\big[\underline{F}_t^k\big]^\star \geq V_t$.*

Proof. The sequence of functions $\underline{F}_t^k$ is obtained by applying SDDP to the dual recursion (3.8), which is an increasing sequence of lower approximations of the function F_t by Proposition 2.15. By the conjugacy property, we obtain a decreasing sequence of functions $\big[\underline{F}_t^k\big]^\star$ which are upper approximations of the function $[F_t]^\star = V_t$. ∎

[5] Note that $\boldsymbol{\mu}_5$ and $\boldsymbol{\mu}_6$ are *constant* random variables.

Algorithm 3.2 Dual SDDP algorithm.

Data: Initial primal point x_0, Lipschitz bounds $\left\{L_t\right\}_{t\in[\![0,T]\!]}$
$\underline{F}_t^0 \leftarrow -\infty$
for $k : 0,\ 1, \dots$ **do**
  // Forward Pass : compute a set of trial points $\left\{\lambda_t^{(k)}\right\}_{t\in[\![0,T]\!]}$
  Select $\lambda_0^k \in \arg\max_{\|\lambda_0\|_\infty \le L_0} \left\{x_0^\top \lambda_0 - \underline{F}_0^k(\lambda_0)\right\}$ // Fenchel transform
  for $t : 0$ **to** $T-1$ **do**
    select $\boldsymbol{\lambda}_{t+1}^k \in \arg\min \mathcal{T}^\sharp_{t,L_{t+1}}(\underline{F}_{t+1}^k)(\lambda_t^k)$
    draw a realization λ_{t+1}^k of $\boldsymbol{\lambda}_{t+1}^k$
  end for
  // Backward Pass : refine the lower-approximations at the trial points
  $\underline{F}_T^k \leftarrow K^\star$.
  for $t : T-1$ **to** 0 **do**
    $\overline{\theta}_t^{k+1} \leftarrow \mathcal{T}^\sharp_{t,L_{t+1}}(\underline{F}_{t+1}^k)(\lambda_t^k)$; select $\overline{x}_t^{k+1} \in \partial\Big[\mathcal{T}^\sharp_{t,L_{t+1}}(\underline{F}_{t+1}^k)\Big](\lambda_t^k)$ // computing cut
    $\overline{\beta}_t^{k+1} \leftarrow \overline{\theta}_t^{k+1} - \left\langle \lambda_t^k, \overline{x}_t^{k+1}\right\rangle$
    $\underline{F}_t^{k+1} \leftarrow \max\left\{\underline{F}_t^k, \left\langle \overline{x}_t^{k+1}, \cdot\right\rangle + \overline{\beta}_t^{k+1}\right\}$ // update lower approximation
  end for
  STOP if some stopping test is satisfied
end for

We have the following convergence theorem.

THEOREM 3.10. *Under Assumptions* 1.1 *and* 3.1, $\left[\underline{F}_0^k\right]^\star(x_0)$ *is a converging upper bound to the value* $V(x_0)$ *of problem* (1.1), *that is,* $\lim_{k\to+\infty}\left[\underline{F}_0^k\right]^\star(x_0) = V_0(x_0)$.

Proof. We add a dummy time step $t = -1$ which allows for a varying initial state, a feature not included in the abstract SDDP algorithm. We are thus able to compute the Fenchel transform of $\underline{F}_0^k$ at x_0. More precisely, we define $\mathcal{T}^\sharp_{-1,L_0}$ as

$$\mathcal{T}^\sharp_{-1,L_0}(R) := \min_{\|\lambda_0\|_\infty \le L_0} -x_0^\top \lambda_0 + R(\lambda_0)\ .$$

Then Algorithm 3.2 is the abstract SDDP Algorithm 2.1 applied to the dual Bellman recursion $F_T = [K]^\star$ and $F_t = \mathcal{T}^\sharp_{t,L_{t+1}}(F_{t+1})$ for $t \in [\![-1, T-1]\!]$. The initial point λ_{-1} is arbitrarily set to the value 0 as F_{-1} is the constant function.

We check that $\|\lambda_t^k\|_\infty \le L_t$ by definition of $\mathcal{T}^\sharp_{t,L_t}$. Furthermore, as V_0 is L_0-Lipschitz for the L_1-norm, the supremum of $x_0^\top\lambda - [V_0]^\star(\lambda)$ is attained for some λ_0 such that $\|\lambda_0\|_\infty \le L_0$; thus we deduce that $F_{-1}(0) = -[V]^{\star\star}(x_0) = -V(x_0) \in \mathbb{R}$. Finally, note that $\left\{\mathcal{T}^\sharp_{t,L_{t+1}}\right\}_{t\in[\![-1,T]\!]}$ is a $[K]^\star$-compatible sequence of LBOs.

Lemma 3.9 shows that, for any $k \in \mathbb{N}$, $-\underline{F}_{-1}^k(0) = \left[\underline{F}_0^k\right]^\star(x_0)$ is an upper bound of $V_0(x_0)$. Finally, the convergence of the abstract SDDP algorithm and the lower semicontinuity of V_0 at x_0 yields the convergence of the upper bound. ∎

Remark 3.11. Recall that in order to obtain an upper bound of the optimal value of problem (1.1), the seminal method consists in computing the expected cost of the SDDP algorithm's strategy with a Monte Carlo approach (see the discussion in subsection 1.2). This approach has two weaknesses: it requires a large number M of forward pass (simulation), and the obtained bound is only an upper bound with

(asymptotic) probability α, where the bound increases with α as well. Furthermore, the statistical upper bound is not converging toward the actual problem value, unless we also increase the number of Monte Carlo simulations along the iterations.

In contrast to the Monte Carlo method, Theorem 3.10 shows that we obtain a converging sequence of exact upper bounds for problem (1.1).

Remark 3.12. In the forward pass of Algorithm 3.2, we need to solve $\mathcal{T}^{\sharp}_{t,L_{t+1}}(\underline{F}^k_{t+1})(\lambda^k_t)$. Note that drawing a realization of the random variable $\boldsymbol{\lambda}^k_{t+1}$ consists of drawing ξ with respect to the probability law of $\boldsymbol{\xi}_{t+1}$, and then selecting the associated value $\lambda^{k,\xi}_{t+1}$. In contrast with the primal SDDP algorithm (see Remark 3.3), here we need to solve a linear problem coupling all possible outcomes of the random variable $\boldsymbol{\xi}_{t+1}$, both in the forward and in the backward pass. In particular it means that we can also compute cuts during the forward pass, thus rendering the backward pass optional.

4. Inner-approximation strategy. In section 3, we detailed how to use the SDDP algorithm to get dual outer approximations $\left\{\underline{F}_t\right\}_{t\in[\![0,T]\!]}$ of the dual value functions $\left\{F_t\right\}_{t\in[\![0,T]\!]}$. We now explain how to build inner approximations of the primal value functions $\left\{V_t\right\}_{t\in[\![0,T]\!]}$ using these dual outer approximations. We assume that the real sequence $\left\{L_t\right\}_{t\in[\![0,T-1]\!]}$ is given by Lemma 2.14.

4.1. Inner approximation of value functions. Let $\left\{\underline{F}^k_t\right\}_{t\in[\![0,T]\!]}$ be the outer approximations of the dual value functions $\left\{F_t\right\}_{t\in[\![0,T]\!]}$ obtained at iteration k of the dual SDDP algorithm. We denote by $\left\{(\overline{x}^\kappa_t,\overline{\beta}^\kappa_t)\right\}_{\kappa\in[\![1,k]\!]}$ the coefficients of the cuts computed by the dual SDDP algorithm: $\underline{F}^k_t(\lambda)=\max_{\kappa\le k}\left\langle \overline{x}^\kappa_t\,,\lambda\right\rangle+\overline{\beta}^\kappa_t$.

We obtain the inner approximations $\left\{\overline{V}^k_t\right\}_{t\in[\![0,T]\!]}$ of the primal value functions $\left\{V_t\right\}_{t\in[\![0,T]\!]}$ by Lipschitz regularization of the Fenchel conjugates of the dual outer approximations.

DEFINITION 4.1. *The inner approximation* $\overline{V}^k_t$ *of* V_t *is defined by*

$$\overline{V}^k_t=\left[\underline{F}^k_t\right]^\star\square\left(L_t\|\cdot\|_1\right)\qquad\forall t\in[\![0,T]\!]\,. \tag{4.1}$$

By definition of Lipschitz regularization, we have the following properties of the functions $\overline{V}^k_t$. The proof is given in Appendix C.

PROPOSITION 4.2. *For any* $t\in[\![0,T-1]\!]$*, the following properties hold true.*

(i) $\overline{V}^k_t\ge V_t$ *on* $\operatorname{dom}\mathcal{G}_t$.

(ii) *The inner approximation* $\overline{V}^k_t$ *is such that*

$$\overline{V}^k_t(x)=\min_{y\in\mathbb{R}^{n_x},\sigma\in\Delta}\left\{L_t\|x-y\|_1-\sum_{\kappa=1}^k\sigma_\kappa\overline{\beta}^\kappa_t\;\middle|\;\sum_{\kappa=1}^k\sigma_\kappa\overline{x}^\kappa_t=y\right\}, \tag{4.2}$$

where $\Delta=\left\{\sigma\in\mathbb{R}^k\mid\sigma\ge0\,,\ \sum_{\kappa=1}^k\sigma_\kappa=1\right\}$ *is the simplex of* $\mathbb{R}^k$.

(iii) *The inner approximation* $\overline{V}^k_t$ *can be computed by solving*

$$\overline{V}^k_t(x)=\sup_{\lambda,\theta}\quad x^\top\lambda-\theta \tag{4.3a}$$

$$s.t.\quad\theta\ge\left\langle\overline{x}^i_t\,,\lambda\right\rangle+\overline{\beta}^\kappa_t\qquad\forall\kappa\in[\![1,k]\!]\,, \tag{4.3b}$$

$$\|\lambda\|_\infty\le L_t\,. \tag{4.3c}$$

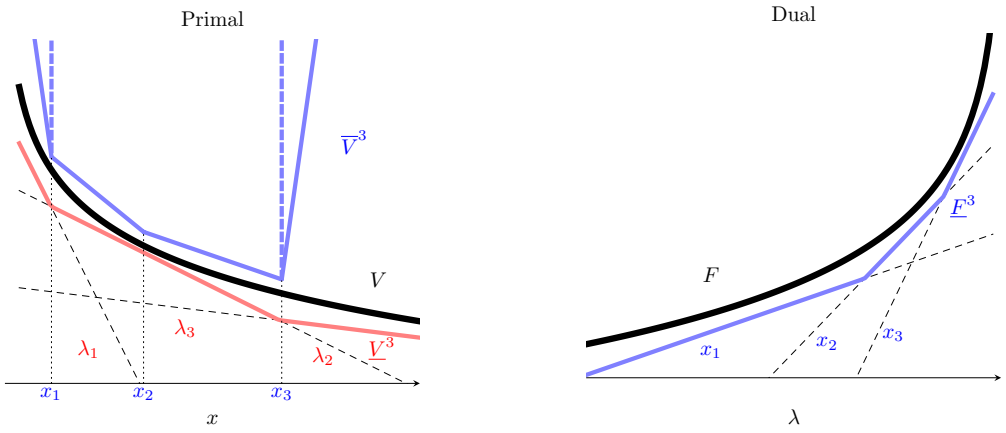


FIG. 1. *On the left we represent the primal value function V as a function of x, while on the right we represent the dual value function F as a function of λ (black curves). Primal SDDP computes an outer approximation $\underline{V}^k$ (in red, left) of V. Dual SDDP computes an outer approximation $\underline{F}^k$ (in blue, right) of F. The (regularized) Fenchel transform of this dual outer approximation is $\overline{V}^k$, an inner approximation of V (in blue, left). Note that the breakpoints of this inner approximation are given by the slopes of the dual outer approximation (in blue, right). (Color available online.)*

(iv) *The Fenchel transform of the inner approximation $\overline{V}_t^k$ is given by $\left[\overline{V}_t^k\right]^\star = \underline{F}_t^k + \mathbb{I}_{B_\infty(0,L_t)}$.*

Figure 1 illustrates how to use the dual outer approximation of F_t to obtain a primal inner approximation of the original value function V_t.

Remark 4.3. Consider a value function following $V_t = \mathcal{B}_t(V_{t+1})$. Assuming that we have an inner bound $\overline{V}_{t+1} \geq V_{t+1}$, a primal way of computing an upper bound of $V_t(x)$ consists simply in computing $v = \mathcal{B}_t(\overline{V}_{t+1})(x)$. If $\overline{V}_{t+1} = \left[\underline{F}_{t+1}\right]^\star$, this is equivalent to computing the Fenchel transform of the outer approximation of the dual:

$$\mathcal{B}_t(\overline{V}_{t+1})(x) = \left[\mathcal{B}_t(\overline{V}_{t+1})\right]^{\star\star}(x) = \left[\mathcal{B}_t^\ddagger(\left[\overline{V}_{t+1}\right]^\star)\right]^\star(x) = \left[\mathcal{B}_t^\ddagger(\underline{F}_{t+1})\right]^\star(x).$$

This approach was used in [17, 1, 9] to construct inner approximation, the main difference being in the choice of the primal point x at which to compute the upper bound.

4.2. A bound on the inner approximation strategy value. Hence, we have obtained inner approximations of the primal value functions. Such approximations can be used to define an admissible strategy for the initial problem. We now study the properties of such a strategy. The proof of the following theorem is given in Appendix C.

THEOREM 4.4. *Let $\left\{\boldsymbol{U}_t^{IA,k}\right\}_{t\in[\![1,T]\!]}$ be the strategy induced by the inner approximations $\overline{V}_t^k$, and let $\left\{\boldsymbol{X}_t^{IA,k}\right\}_{t\in[\![0,T]\!]}$ be the associated state process, that is, $\boldsymbol{X}_{t+1}^{IA,k} \in \mathcal{S}_t(\overline{V}_{t+1}^k)(\boldsymbol{X}_t^{IA,k})$. Consider the expected cost of this strategy when starting from state x at time t:*

$$C_t^{IA,k}(x) = \mathbb{E}\left[\sum_{\tau=t}^{T-1}\left(a_\tau^\top \boldsymbol{X}_\tau^{IA,k} + b_{\tau+1}^\top \boldsymbol{U}_{\tau+1}^{IA,k}\right) + K(\boldsymbol{X}_T^{IA,k}) \;\middle|\; \boldsymbol{X}_t^{IA,k} = x\right].$$

Then

$$C_t^{IA,k}(x) \leq \overline{V}_t^k(x)\ . \tag{4.4}$$

Furthermore, the strategy induced by the inner approximations is converging in the sense that $\lim_{k\to+\infty} C_0^{IA,k}(x_0) = V_0(x_0)$.

Remark 4.5. The proposed inner approximation differs from the literature [1, 17] mainly by relying on the dual formulation of the problem. In [17] the authors derive an inner approximation from a given set of trajectories. They suggest using the forward trajectories of a primal SDDP but, contrary to our approach, they do not show any convergence results of the upper bound. Furthermore, computing the upper bound is quite time consuming, and computation at iteration k cannot be used for the computation at iteration $k+1$, whereas computing our bound from dual SDDP is fast.

In [1] the authors obtain a converging deterministic upper bound. However, instead of sampling the noise in the forward phase, they select the random realization leading to a state with the highest gap between lower and upper bound. This has the advantage of having a fully deterministic algorithm which can sometimes get stuck. In particular our approach should be more efficient if the support of $\boldsymbol{\xi}_t$ is large with the probability mass concentrated around a few elements (e.g., binomial random variables).

We sum up the available inequalities for the values obtained when implementing the primal and dual SDDP algorithms (iteration index k is omitted for the sake of clarity):

$$\underline{V}_0(x_0) \ \leq\ V_0(x_0) \ \leq\ \overline{V}_0(x_0)\ , \tag{4.5a}$$

$$\underline{V}_0(x_0) \ \leq\ C_0^{IA}(x_0) \ \leq\ \overline{V}_0(x_0)\ , \tag{4.5b}$$

$$\underline{V}_0(x_0) \ \leq\ C_0^{OA}(x_0)\ . \tag{4.5c}$$

Equation (4.5a) shows that $\underline{V}_0(x_0)$ and $\overline{V}_0(x_0)$ are deterministic bounds on the value of problem (1.1). Equation (4.5b) shows that $\overline{V}_0(x_0)$ is also an upper bound on the expected value of the strategy induced by the inner approximation. Unfortunately, as we can see in Figure 2, there is no generic relation between $C_0^{OA}(x_0)$ and $\overline{V}_0(x_0)$, except that both converge toward the true value of the problem.

5. Numerical results. In this section, we present some numerical results obtained when applying dual SDDP and inner strategy evaluation to a stochastic operation planning problem inspired by Électricité de France (EDF, one of the main European electricity producers). The problem is concerned with energy production planning on a multiperiod horizon including a network of production zones, like in the European Electricity Market. This results in a large-scale stochastic multistage optimization problem, for which we need to determine strategies for the management of the European water dams. Such strategies cannot be computed via dynamic programming because of the state variable size, so that SDDP is the reference method to compute the optimal Bellman functions.

5.1. Description of the problem. We consider an operation planning problem at the European scale. Different countries are connected together via a network and exchange energy with their neighbors. We formulate the problem on a graph where each country is modeled as a node and each interconnecting line between two countries as an edge.

Every country uses a reservoir to store energy and must fulfill its own energy demand. To do so, it can produce energy from its reservoir, with its local thermal power plant, or it can import energy from the other countries. A very similar problem has already been studied by [15]. Its formulation is close to the one given in [23] concerning the Brazilian interconnected power system.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be the graph modeling the European network. The number of nodes in $\mathcal{N}$ is denoted by n and the number of edges in $\mathcal{E}$ by ℓ. For each node $i \in [\![1, n]\!]$, we denote by v_t^i the energy stored in the reservoir at time t. The reservoir's dynamics is given by

$$v_{t+1}^i = v_t^i + a_{t+1}^i - q_{t+1}^i - s_{t+1}^i \ , \tag{5.1}$$

where a_{t+1}^i is the (random) water inflow in the reservoir and q_{t+1}^i is the water flowing through a turbine between time t and $t+1$ in order to produce electricity. We add a spillage s_{t+1}^i as a recourse variable to avoid reservoir overflow. Still at node i, the *load balance* equation at stage t is written as

$$q_t^i + g_t^i + \sum_{j \in N_i} f_t^{ji} + r_t^i = d_t^i \ , \tag{5.2}$$

where g_t^i is the thermal production, $N_i \subset \mathcal{N}$ is the set of nodes connected to node i, f_t^{ji} is the energy exchanged between nodes $j \in N_i$ and node i, d_t^i is the (random) demand of the node, and r_t^i is a recourse variable added to ensure that the load balance is always satisfied. The thermal production g_t^i and the exchanges f_t^{ji} between node i and nodes $j \in N_i$ induce linear costs, and the cost of the recourse variable r_t^i is taken into account through a linear penalization. Hence the total cost attached to node i at time t is

$$c_t^i g_t^i + \delta_t^i r_t^i + \sum_{j \in N_i} p_t^{ji} f_t^{ji} \ , \tag{5.3}$$

where c_t^i is the thermal price, δ_t^i is the recourse price, and p_t^{ji} is the transportation price between nodes j and i. To avoid empty stocks at the end of the time horizon, we penalize the final stock at each node i if it is beyond a threshold v_0^i using a piecewise linear function:

$$K^i(v_T^i) = \kappa_T^i \max(0, v_0^i - v_T^i) + \mathbb{I}_{0 \le v_T^i \le \overline{v}^i} \ . \tag{5.4}$$

Stocks and controls are bounded:

- $0 \le v_t^i \le \overline{v}^i$, reservoir volume lower and upper bounds;
- $0 \le q_t^i \le \overline{q}^i$, reservoir generation lower and upper bounds;
- $0 \le g_t^i \le \overline{g}^i$, thermal generation lower and upper bounds;
- $0 \le r_t^i$, recourse control lower bound;
- $\underline{f}^{ji} \le f_t^{ji} \le \overline{f}^{ji}$, energy flow lower and upper bounds.

This problem is formulated as a stochastic optimal control problem, where for all t,

- the state is $v_t = (v_t^1, \dots, v_t^n)$ (denoted x_t in section 3);
- the control is $u_t = (q_t, s_t, g_t, r_t, f_t)$, with $q_t = (q_t^i)_{i \in [\![1,n]\!]}$ and likewise for s_t, g_t, r_t, and f_t;
- the uncertainty is $\xi_t = (a_t^i, d_t^i)_{i \in [\![1,n]\!]}$.

The state has dimension n, the control u_t dimension $4n + \ell$, and the uncertainty ξ_t dimension $2n$. We assume that the random variables $\boldsymbol{\xi}_t$ have a discrete finite support.

5.2. Numerical implementation. The forward and backward passes of dual SDDP are *independent* of the forward and backward passes of primal SDDP. Accordingly, a first "natural" implementation of the whole algorithm runs primal and dual SDDP in two independent processes, and thus enables us to compute primal and dual value functions in parallel.

However, each backward pass of the primal SDDP algorithm computes a set of cuts whose slopes are $\left\{\lambda_t\right\}_{t\in[\![0,T]\!]}$. As explained when commenting on Figure 1, these slopes can be considered as state trajectories for the dual problem. When primal SDDP has converged, these slopes are the optimal costates of the problem, because of the Fenchel–Young equality. Therefore, it may prove useful to use these sequences of slopes as state trajectories for the dual problem, along which we run afterward a backward pass producing cuts for the dual problem. In this implementation, each iteration of the algorithm consists of four steps:

1. Run a forward pass of primal SDDP Algorithm 3.1 and get a state trajectory $\left\{x_t\right\}_{t\in[\![0,T]\!]}$.
2. Run a backward pass of primal SDDP Algorithm 3.1 along $\left\{x_t\right\}_{t\in[\![0,T]\!]}$ and obtain new slopes $\left\{\lambda_t\right\}_{t\in[\![0,T]\!]}$.
3. Run a backward pass of dual SDDP Algorithm 3.2 along $\left\{\lambda_t\right\}_{t\in[\![0,T]\!]}$ and obtain new cuts for the dual problem.
4. Run a forward pass of dual SDDP Algorithm 3.2, obtain a trajectory, and simultaneously update the cuts along this trajectory.

The last step of this iteration ensures that we recover the convergence assumptions of SDDP (as given in [10]).

This algorithm has the same number of forward and backward passes as the original one (one forward pass and one backward pass in both primal and dual spaces). However, this scheme proves to be numerically more efficient, in terms of both convergence and computation time. That is why we use this implementation in all the numerical experiments. Furthermore, in our example, this scheme is quite resilient to an overestimation of the Lipschitz constant.

From the computational point of view, we implement primal and dual SDDP in Julia 0.6, with the `StochDynamicProgramming.jl` package built on top of the JuMP modeler of [8]. We use Gurobi 7.02 to solve the linear programming subproblems. All experiments are run on an Intel Core i7-5500 CPU @2.4GHz, 64-bit computer. The source code is currently available on Github.[6]

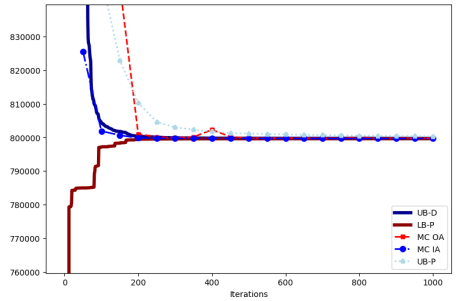
5.3. Results. We consider the problem described at subsection 5.1, with $n = 8$ and $T = 12$ or $T = 36$. We aim to compute the value functions $\left\{V_t\right\}_{t\in[\![0,T]\!]}$ with monthly time steps. The uncertainties in the model are the inflows a_t in the reservoir and the demands d_t in every considered countries. Inflow and demand trajectories are simulated using software provided by EDF, so that these data are realistic enough. From these simulated samples, we use quantization methods to obtain the marginal laws of the uncertainty $\boldsymbol{\xi}_t$ at each $t \in [\![0, T]\!]$. The support of the quantized probability laws is limited to 10 possible values for $\boldsymbol{\xi}_t$ at each time step t. To solve the problem, we run primal and dual SDDP on 1,000 iterations, with a single forward pass in the primal and in the dual.

5.3.1. Assessing convergence. To simplify the description of the results, we denote by `LB-P` the primal lower bound $\underline{V}_0(x_0)$ obtained by primal SDDP, and by `UB-D`

[6] https://github.com/frapac/DualSDDP.jl

the upper bound $[\underline{F}_0]^\star(x_0)$ given by dual SDDP. The Monte Carlo cost evaluation obtained by simulating the outer (resp., inner) strategy uses the procedure described in subsection 3.1.2 and is denoted by `MC OA` (resp., `MC IA`). Confidence intervals (with a confidence level $\alpha = 97.5\%$) are associated with these Monte Carlo approximations, and we denote by `MC OA UB` and `MC IA UB` the associated upper bounds of these intervals. Whereas `LB-P` and `UB-D` are deterministic values, `MC OA`, `MC IA`, `MC OA UB`, and `MC IA UB` are statistical quantities.

Solving the problem over a one-year time horizon. First, we run dual and primal SDDP on a twelve month problem, that is, with $T = 12$. Convergence of the optimal costs given by dual and primal SDDP is detailed in Figure 2.



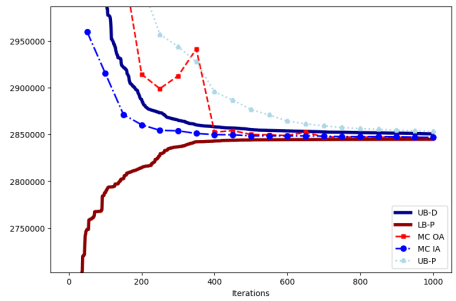
Iter.	LB-P	UB-D	UB-P	Gap	LB-P	UB-D
Unit	$\times 10^5$	$\times 10^5$	$\times 10^5$	%	s	s
50	7.850	11.37	18.58	44.8	2.	8.
100	7.971	8.046	8.475	9.43	3.	22.
200	7.994	7.999	8.103	1.20	8.	72.
300	7.996	7.998	8.030	0.40	13.	153.
400	7.996	7.998	8.017	0.23	20.	275.
500	7.996	7.997	8.012	0.17	29.	443.
600	7.996	7.997	8.009	0.14	38.	651.
700	7.996	7.997	8.007	0.12	49.	888.
800	7.996	7.997	8.005	0.11	61.	1191.
900	7.996	7.997	8.004	0.10	74.	1534.
1000	7.996	7.997	8.003	0.09	89.	1928.

FIG. 2. *Convergence of primal and dual SDDP for $T = 12$. Time corresponds to cumulated time along iterations.*

In the table of Figure 2, UB-D is the upper bound obtained by dual SDDP, while UB-P is the upper bound obtained as in [17] using the forward passes of primal SDDP. We note that UB-D is slightly better. More importantly it is easily computed every few iterations, while computing UB-P is more time consuming. The two last columns in the table give the cumulative computation times needed to run both primal and dual SDDP algorithms. We observe that the upper bound `UB-D` $[\underline{F}_0]^\star(x_0)$ given by dual SDDP converges towards the primal lower bound `LB-P` $\underline{V}_0(x_0)$ given by primal SDDP, with a relative gap close to 0.09% after 1,000 iterations. For this specific (with few time steps) example, the convergence of dual SDDP proves to be effective and the dual upper bound beat the primal exact upper bound. As noticed at Remark 3.12, running dual SDDP is much more time consuming than running primal SDDP.

The outer and inner strategies are evaluated by Monte Carlo. We draw a "large" set of 10,000 scenarios on which evaluation is performed every 50 iterations. Both evaluations `MC OA` and `MC IA` converge to the optimal value. We notice that `MC IA` is below `UB-D`, thus illustrating the result stated by Theorem 4.4.

Solving the problem over a three-year time horizon. We now consider the same problem, but over a three-year horizon, that is, with $T = 36$. The convergence of primal and dual SDDP is shown in Figure 3. We have materialized the confidence intervals (here very thin) of the inner and outer strategies by Monte Carlo simulations, both estimated every 50 iterations on the given 10,000 scenarios. A first observation is that both dual and primal SDDP exhibit a slower convergence than in the first example: after 1,000 iterations, the gap between the primal lower bound `LB-P` and the dual upper bound `UB-D` is equal to 0.20%. This well-known behavior of SDDP arises from the increasing number of time steps (36 instead of 12). Moreover, `UB-D` is still significantly decreasing after iteration 500, and it seems that it converges more slowly than `LB-P`.



Iter.	LB-P	UB-D	UB-P	Gap	LB-P	UB-D
Unit	$\times 10^6$	$\times 10^6$	$\times 10^6$	%	s	s
50	2.748	3.456	8.453	25.76	5.	20.
100	2.788	3.004	4.365	7.75	11.	74.
200	3.814	2.888	3.011	2.61	27.	267.
300	2.837	2.865	2.944	1.01	46.	592.
400	2.843	2.858	2.895	0.54	75.	1113.
500	2.844	2.855	2.877	0.39	108.	1783.
600	2.844	2.854	2.864	0.33	144.	2601.
700	2.844	2.853	2.859	0.29	187.	3585.
800	2.845	2.852	2.856	0.26	235.	4751.
900	2.845	2.851	2.854	0.23	296.	6140.
1000	2.845	2.850	2.853	0.20	360.	7545.

FIG. 3. *Convergence of primal and dual SDDP for $T = 36$. Time corresponds to cumulated time along iterations.*

A second observation is that the dual upper bound `UB-D` is better than the statistical cost value `MC OA` up to iteration 400. After the first 500 iterations, `MC OA` is better than `UB-D` and fluctuates slightly above the primal lower bound `LB-P` (the remaining gap being around 0.1% after 1,000 iterations).

Finally, on this example, the inner approximation performs better than the outer approximation by yielding a better value for most iterations and displaying a more stable convergence. It would be interesting to be able to assess such behavior.

5.3.2. Using the dual upper bound in a stopping criterion. Consider the problem over a three-year time horizon. The gap between the two deterministic bounds (primal lower bound `LB-P` and dual upper bound `UB-D`) against the number of iterations is given in Figure 3. To complete these results, we give the evolution of the statistical upper bound `MC OA UB` obtained by the outer strategy in Table 1. We aim at comparing two stopping tests.

TABLE 1
Statistical upper bound for $T = 36$.

Iter.	`LB-P`	Gap `UB-D` (%)	`MC OA UB`	Gap `MC OA UB` (%)
50	2.837	38.1	3.392	19.6
100	2.980	5.7	3.310	11.1
200	3.029	1.4	3.137	3.6
300	3.039	0.67	3.069	1.0
400	3.040	0.46	3.059	0.62
500	3.041	0.34	3.046	0.18
600	3.041	0.25	3.046	0.18
700	3.041	0.21	3.046	0.15
800	3.041	0.18	3.046	0.16
900	3.041	0.16	3.045	0.14
1000	3.041	0.13	3.044	0.08

Statistical stopping test: This is the stopping test proposed in [21] and which has been detailed in subsection 1.2. We choose a confidence level $\alpha = .975$, and we estimate the statistical upper bound `MC OA UB` every 50 iterations with a given set of 10,000 scenarios.

Dual stopping test: This stopping test is just based on the gap between the available deterministic upper and lower bounds, namely, `UB-D` and `LB-P`.

For different accuracy levels ε, as described by [21] we compare the CPU times taken by these two tests in order to stop the SDDP algorithm. Results are given in Table 2.

TABLE 2
Dual and statistical stopping criteria for different accuracy levels ε.

	Dual		Statistical	
ε (%)	n iter.	CPU	n iter.	CPU
2.0	156	183s	250	618s
1.0	236	400s	300	787s
0.5	388	1116s	450	1429s
0.1	> 1000	.	1000	5519s

The given times correspond to the total time required to run SDDP (including both the computation of cuts and the computation of the stopping test). We notice that the dual stopping test gives better results than the statistical stopping test: for $\varepsilon \geq 0.45\%$, it stops SDDP earlier and require less computation time. Compared with the statistical test, the speed-up is between 3.3 for $\varepsilon = 2\%$ and 1.3 for $\varepsilon = 0.5\%$. However, the dual stopping test is penalized by the slow convergence of dual SDDP. Indeed it cannot achieve a gap lower than 0.1 %, thus penalizing the performance of the dual stopping test for high accuracy levels ε.

As a conclusion of these numerical experiments, the deterministic dual stopping test seems to be better than the statistical stopping test, especially if restrictions on the CPU time impose performing a limited number of SDDP iterations (less than 500 in our case). Such a situation exists in the energy field, as shown by the description of the Brazilian interconnected power system in [23]. A way to significantly reduce the computation time of dual SDDP (as well as that of primal SDDP) would be to implement a cut selection mechanism in the algorithm, which would limit the number of constraints added to the problem; see, for example, [7] and [12] for further details.

Remark 5.1. We can also use the statistical upper bound `MC IA UB` obtained by evaluating the inner strategy for statistical stopping tests. Indeed, in our numerical experiments, this upper bound is always lower than the one given by the outer strategy. However, this would require much longer computational time, as this approach combines the computation of the dual cuts together with a Monte Carlo estimation.

Remark 5.2. We observe that the convergence of dual SDDP is penalized by different considerations.

- It is well known from [21] that the convergence of SDDP highly depends on the number of stages in the problem, as well as the size of the state vectors. This issue impacts both primal and dual SDDP.
- Furthermore, we notice that dual SDDP exhibits a slower convergence than primal SDDP. In fact, primal SDDP computes its trajectories from a fixed initial point x_0, whereas, as explained at subsection 3.2.3, dual SDDP updates its initial point λ_0 at each iteration, with

$$\lambda_0^k \in \underset{\|\lambda_0\| \leq L_0}{\arg\min} -x_0^\top \lambda_0 + \underline{F}_0^k(\lambda_0) \,. \tag{5.5}$$

- One iteration of dual SDDP takes longer than one iteration of primal SDDP. Indeed, as already pointed out in Remarks 3.3 and 3.12, dual SDDP solves bigger linear programming problems than primal SDDP, as it has to take into account a coupling constraint between all samples.

6. Conclusion. In this paper, we have shown that dual SDDP allows us to obtain a deterministic stopping criterion which proves to be effective compared to the standard statistical stopping test. This stopping criterion uses an exact upper

bound, the computation of which relies on applying SDDP to the Fenchel transform of Bellman value functions. Furthermore, dual SDDP computes cuts that can be used to design an inner approximation strategy, which appears to be better than the outer strategy as long as primal SDDP has not exactly converged. The method provides a sequence of inner approximations $\left\{\overline{V}_t\right\}_{t\in[\![0,T]\!]}$ of the primal value functions $\left\{V_t\right\}_{t\in[\![0,T]\!]}$. The policy induced by inner approximations converges to an optimal policy, with guaranteed performance of the associated expected cost.

We have tested dual SDDP and presented numerical results on a realistic stochastic production planning problem, proving the effectiveness of dual SDDP and the underlying inner strategy. Furthermore, we showed on this particular problem that using a dual stopping test outperforms the classical statistical stopping test of SDDP, in terms of both the number of iterations and the computational burden.

We plan to extend this study in several directions. First, an extension of dual SDDP to risk-averse or distributionally robust problems remains to be investigated. Second, the dual SDDP algorithm does not decompose the computation of the dual LBOs by realizations of $\boldsymbol{\xi}_t$. A means of effectively decomposing the dual subproblems is still under study. Finally we want to explore the interactions between primal and dual SDDP. For example, we think that the upper bounds given by dual SDDP might be effective in regularizing SDDP, for instance, with the method introduced by [24].

Appendix A. LBO related proofs.

Proof of Proposition 2.6. The probability set Ω being finite, we denote by u (resp., y, c, h) the vectors concatenating all possible values of $\boldsymbol{U}$ (resp., $\boldsymbol{Y}$, $\boldsymbol{C}$, $\boldsymbol{H}$) over the set Ω, that is, $u = (u_1, \dots, u_{|\Omega|})$. Then the extensive formulation of constraint (2.2) is $\widetilde{T}x + \widetilde{W}_u u + \widetilde{W}_y y \leq h$, where $\widetilde{T}$, $\widetilde{W}_u$, and $\widetilde{W}_y$ are adequate matrices deduced from T, $\mathcal{W}_u$, and $\mathcal{W}_y$. Problem (2.1) rewrites $\mathcal{B}(R)(x) = \inf_{u,y}\ J(R)(x,u,y)$, with

$$J(R)(x,u,y) = \sum_{\omega=1}^{|\Omega|} \pi_\omega \Big(c_\omega^\top u_\omega + R(y_\omega) \Big) + \mathbb{I}_{\{\widetilde{T}x + \widetilde{W}_u u + \widetilde{W}_y y \leq h\}}(x,u,y)\ .$$

1. If R is convex, then $J(R)$ is jointly convex in (x,u,y), so that $\mathcal{B}(R)$ is a convex function.
2. If R is polyhedral, then $J(R)$ is polyhedral in (x,u,y), and thus $\mathcal{B}(R)$ is a polyhedral function (see [6, Proposition 5.1.8]).
3. From $R \geq \tilde{R}$, we deduce that $J(R) \geq J(\tilde{R})$, and thus $\mathcal{B}(R) \geq \mathcal{B}(\tilde{R})$.

The proof is complete. ∎

Appendix B. SDDP related proofs.

Proof of Lemma 2.14. We prove by induction that the points $x_t^{(k)}$ are well defined during the forward passes of SDDP. Let $t = 0$. By assumption, $x_0^k \in \mathrm{dom}(\mathcal{G}_0)$. So $x_1^k = \boldsymbol{X}_1^{(k)}(\omega^k)$ exists as a solution of a finite-valued linear program. Let $t \geq 1$. By the induction hypothesis, we suppose that x_t^k is well defined and belongs to $\mathrm{dom}(\mathcal{G}_t)$. We set $x_{t+1}^k = \boldsymbol{X}_{t+1}^{(k)}(\omega^k)$, which is well defined as a solution of a finite-valued linear program. By assumption the sequence $\left\{\mathcal{B}_t\right\}_{t\in[\![0,T-1]\!]}$ is K-compatible, and hence $x_{t+1}^k \in \mathrm{dom}(\mathcal{G}_{t+1})$, thus proving the result at time $t+1$.

We now prove by backward induction that $\lambda_t^{(k)}$ is well defined during the backward passes of SDDP, and that there exists L_t such that $\|\lambda_t^{(k)}\|_\infty \leq L_t$. As K is a given

L_T Lipschitz-continuous function, the property holds true for $t = T$. Let $t \leq T-1$. Assume that the induction hypothesis holds for $t+1$. Then, by Proposition 2.8, we know that $\mathcal{B}_t(\underline{R}_{t+1}^{k+1})$ is L_t-Lipschitz. We set $\lambda_t^{k+1} \in \partial\mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$, which is well defined as subgradient of a finite-valued polyhedral function. As $\mathcal{B}_t(\underline{R}_{t+1}^{k+1})$ is L_t-Lipschitz on its domain, we are able to choose λ_t^{k+1} such that $\|\lambda_t^{(k+1)}\|_\infty \leq L_t$. ∎

Proof of Lemma 3.2. As problem (1.1) is finite valued, the domain of each set-valued mapping $\mathcal{G}_t$ is nonempty. Further, as $\mathcal{G}_t$ is compact valued with compact domain, each LBO $\mathcal{T}_t$ is compact (see Definition 2.1). Then the reformulation as (3.4) and (3.5) is a direct consequence of the measurability properties of the pair $(\boldsymbol{U}_{t+1}, \boldsymbol{X}_{t+1})$ allowing the interchange between minimization and expectation. ∎

Appendix C. Inner approximation related proofs.

Proof of Proposition 4.2. Let X_t be $\mathrm{dom}(\mathcal{G}_t)$.

(i) Lemma 3.9 proves that $\left[\underline{F}_t^k\right]^\star \geq V_t$ for all $t \in [\![0,T]\!]$. Thus $\overline{V}_t^k \geq V_t \square (L_t\|\cdot\|_1)$, which is equal to V_t on X_t as V_t is L_t-Lipschitz on its domain.

(ii) Furthermore, the Fenchel conjugate $\left[\underline{F}_t^k\right]^\star$ reads

$$\left[\underline{F}_t^k\right]^\star(x) = \sup_{\lambda,\theta} \left\{ x^\top\lambda - \theta \quad | \quad \theta \geq \left\langle \overline{x}_t^i, \lambda \right\rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in [\![1,k]\!] \right\},$$

which is a linear program admitting an admissible solution; hence by strong duality

$$\left[\underline{F}_t^k\right]^\star(x) = \min_{\sigma\in\Delta} \left\{ -\sum_{\kappa=1}^k \sigma_\kappa \overline{\beta}_t^\kappa \quad \Big| \quad \sum_{\kappa=1}^k \sigma_\kappa \overline{x}_t^\kappa = x \right\}.$$

Taking the inf-convolution with $L_t\|\cdot\|$ yields problem (4.2).

(iii) The right-hand side of (4.3) is simply $\left[\underline{F}_t^k + \mathbb{I}_{B_\infty(0,L_t)}\right]^\star(x)$, which is equal to $\left[\underline{F}_t^k\right]^\star \square \left[\mathbb{I}_{B_\infty(0,L_t)}\right]^\star(x)$ by finite polyhedrality, hence the result.

(iv) Finally, $\left[\overline{V}_t^k\right]^\star = \left[\underline{F}_t^k\right]^{\star\star} + \mathbb{I}_{B_\infty(0,L_t)}$. ∎

LEMMA C.1. *We have, for all* $t \in [\![0,T]\!]$, $\mathcal{T}_{t,L}^\ddagger\big(\underline{F}_{t+1}^k\big) \geq \underline{F}_t^k$.

Proof. The relation is satisfied for $k = 0$. Assume that this holds true at iteration k. On the one hand, by definition of $\mathcal{C}^{k+1}$ in Algorithm 3.2, we have $\mathcal{C}^{k+1} \leq \mathcal{T}_{t,L_{t+1}}^\ddagger(\underline{F}_{t+1}^{k+1})$. On the other hand, by monotonicity of $\mathcal{T}_{t,L_{t+1}}^\ddagger$, since $\underline{F}_{t+1}^{k+1} \geq \underline{F}_{t+1}^k$, we have $\mathcal{T}_{t,L_{t+1}}^\ddagger\big(\underline{F}_{t+1}^{k+1}\big) \geq \mathcal{T}_{t,L_{t+1}}^\ddagger\big(\underline{F}_{t+1}^k\big)$, which is greater than $\underline{F}_t^k$ by the induction hypothesis. Thus, $\mathcal{T}_{t,L_{t+1}}^\ddagger\big(\underline{F}_{t+1}^{k+1}\big) \geq \max\left\{\underline{F}_t^k, \mathcal{C}^{k+1}\right\} = \underline{F}_t^{k+1}$. ∎

LEMMA C.2. *Let* $\overline{V}_t^k$ *be the inner approximation of the value function* V_t *generated at iteration* k *of the dual SDDP algorithm. Then, for all* $t \in [\![0,T]\!]$, *we have* $\mathcal{T}_t(\overline{V}_{t+1}^k)(x) \leq \overline{V}_t^k(x)$.

Proof. We have

$$\begin{aligned}
\left[\mathcal{T}_t(\overline{V}_{t+1}^k)\right]^\star &= \mathcal{T}_t^\ddagger\Big(\left[\overline{V}_{t+1}^k\right]^\star\Big) && \text{(by Theorem 2.11)}\\
&= \mathcal{T}_t^\ddagger\big(\underline{F}_{t+1}^k + \mathbb{I}_{B_\infty(0,L_{t+1})}\big) && \text{(by Proposition 4.2)}\\
&= \mathcal{T}_{t,L}^\ddagger\big(\underline{F}_{t+1}^k\big) \geq \underline{F}_t^k. && \text{(by Lemma C.1)}
\end{aligned}$$

Furthermore, as $\mathcal{T}_t(\overline{V}^k_{t+1})$ is polyhedral, we have $\mathcal{T}_t(\overline{V}^k_{t+1}) = \big[\mathcal{T}_t(\overline{V}^k_{t+1})\big]^{\star\star} \leq \big[\underline{F}^k_t\big]^\star$, and as $\overline{V}^k_{t+1}$ is L_{t+1}-Lipschitz, then $\mathcal{T}_t(\overline{V}^k_{t+1})$ is L_t-Lipschitz, and thus $\mathcal{T}_t(\overline{V}^k_{t+1}) \leq \big[\underline{F}^k_t\big]^\star \square (L_t \|\cdot\|)$. ∎

Proof of Theorem 4.4. We proceed by backward induction on time t. The property holds for $t = T$. Assume that $C^{IA}_{t+1} \leq \overline{V}^k_{t+1}$. We have

$$\begin{aligned} C^{IA}_t(x) &= \mathbb{E}\big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}^{IA}_{t+1} + C^{IA}_{t+1}(\boldsymbol{X}^{IA}_{t+1})\big] \\ &\leq \mathbb{E}\big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}^{IA}_{t+1} + \overline{V}^k_{t+1}(\boldsymbol{X}^{IA}_{t+1})\big] && \text{(by induction)} \\ &= \mathcal{T}_t\big(\overline{V}^k_{t+1}\big)(x) && \text{(by definition of } \boldsymbol{U}^{IA}_{t+1}) \\ &\leq \overline{V}^k_t(x)\ . && \text{(by Lemma C.2)} \end{aligned}$$

Finally, the convergence of the strategy is easily obtained. By definition of $V_0(x_0)$, we have $C^{IA,k}_0(x_0) \geq V_0(x_0)$. Furthermore, $V_0(x_0) \leq C^{IA,k}_0(x_0) \leq \overline{V}^k_0(x_0)$. By Theorem 3.10, we know that $\lim_k(\overline{V}^k_0)(x_0) = V_0(x_0)$. Hence the result. ∎

Acknowledgments. We would like to acknowledge the helpful comments of the associate editor and anonymous reviewers. This research benefited from the support of the FMJH "Program Gaspard Monge for Optimization and Operations Research and Their Interactions with Data Science" and from the support of EDF.

REFERENCES

[1] R. Baucke, A. Downward, and G. Zakeri, *A deterministic algorithm for solving multistage stochastic programming problems*, Optimization Online, 2017.
[2] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed., Springer, 2017.
[3] R. Bellman, *Dynamic Programming*, Princeton University Press, NJ, 1957.
[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control. Vol.* II, 3rd ed., Athena Scientific, 2007.
[5] A. R. Bo Lincoln, *Relaxing dynamic programming*, IEEE Trans. Automat. Control, 51 (2006), pp. 1249–1260.
[6] J. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer, 2010.
[7] V. L. de Matos, A. B. Philpott, and E. C. Finardi, *Improving the performance of stochastic dual dynamic programming*, J. Comput. Appl. Math., 290 (2015), pp. 196–208.
[8] I. Dunning, J. Huchette, and M. Lubin, *JuMP: A modeling language for mathematical optimization*, SIAM Rev., 59 (2017), pp. 295–320, https://doi.org/10.1137/15M1020575.
[9] A. Georghiou, A. Tsoukalas, and W. Wiesemann, *Robust dual dynamic programming*, Oper. Res., 67 (2019), pp. 813–830.
[10] P. Girardeau, V. Leclére, and A. B. Philpott, *On the convergence of decomposition methods for multistage stochastic convex programs*, Math. Oper. Res., 40 (2014), pp. 130–145.
[11] V. Guigues, *Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs*, SIAM J. Optim., 26 (2016), pp. 2468–2494, https://doi.org/10.1137/140983136.
[12] V. Guigues, *Dual dynamic programming with cut selection: Convergence proof and numerical experiments*, European J. Oper. Res., 258 (2017), pp. 47–57.
[13] V. Guigues, *Inexact cuts in stochastic dual dynamic programming*, SIAM J. Optim., 30 (2020), pp. 407–438, https://doi.org/10.1137/18M1211799.
[14] T. Homem-de Mello, V. L. De Matos, and E. C. Finardi, *Sampling strategies and stopping criteria for stochastic dual dynamic programming: A case study in long-term hydrothermal scheduling*, Energy Syst., 2 (2011), pp. 1–31.
[15] P. Mahey, J. Koko, and A. Lenoir, *Decomposition methods for a spatial model for long-term energy pricing problem*, Math. Methods Oper. Res., 85 (2017), pp. 137–153.
[16] M. V. Pereira and L. M. Pinto, *Multi-stage stochastic optimization applied to energy planning*, Math. Program., 52 (1991), pp. 359–375.

[17] A. Philpott, V. de Matos, and E. Finardi, *On solving multistage stochastic programs with coherent risk measures*, Oper. Res., 61 (2013), pp. 957–970.
[18] A. Philpott and Z. Guan, *On the convergence of stochastic dual dynamic programming and related methods*, Oper. Res. Lett., 36 (2008), pp. 450–455.
[19] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.
[20] R. T. Rockafellar and R. J.-B. Wets, *Stochastic convex programming: Relatively complete recourse and induced feasibility*, SIAM J. Control Optim., 14 (1976), pp. 574–589, https://doi.org/10.1137/0314038.
[21] A. Shapiro, *Analysis of stochastic dual dynamic programming method*, European J. Oper. Research, 209 (2011), pp. 63–72.
[22] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on Stochastic Programming: Modeling and Theory*, SIAM, Philadelphia, 2009.
[23] A. Shapiro, W. Tekaya, J. P. da Costa, and M. P. Soares, *Final Report for Technical Cooperation between Georgia Institute of Technology and ONS–Operador Nacional do Sistema Elétrico*, Georgia Tech. ISyE Report, 2012.
[24] W. Van Ackooij, W. de Oliveira, and Y. Song, *On regularization with normal solutions in decomposition methods for multistage stochastic programming*, Optimization Online, 2017.