Stochastic dual dynamic programming with stagewise-dependent objective uncertainty

Anthony Downward[a,*], Oscar Dowson[a,b], Regan Baucke[a,c]

[a]*Department of Engineering Science, University of Auckland, New Zealand*
[b]*Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL*
[c]*CERMICS at Ecole des Ponts ParisTech, Champs-sur-Marne, France*

Abstract

We present a new algorithm for solving linear multistage stochastic programming problems with objective function coefficients modeled as a stochastic process. This algorithm overcomes the difficulties of existing methods which require discretization. Using an argument based on the finiteness of the set of possible cuts, we prove that the algorithm converges almost surely. Finally, we demonstrate the practical application of the algorithm on a hydro-bidding example with the spot-price modeled as an auto-regressive process.

Keywords: stochastic programming, dynamic programming, decomposition, multistage, SDDP

1. Introduction

In this paper we consider a multistage stochastic programming problem with stagewise dependent stochastic processes in both the right-hand sides and the objective function coefficients of the stage problem. Multistage convex stochastic programming problems are typically solved using a variant of the SDDP algorithm (stochastic dual dynamic programming) [1]. The simplest implementation of this has stagewise independent stochastic processes. The algorithm employs a forward pass, where a sample path is generated from the stochastic process, and a backward pass which refines the approximations of the future cost functions along the sample path from the forward pass.

Stagewise dependent noise is typically modelled by creating additional state variables to keep track of previous noise outcomes. If the noise is on the right-hand sides of constraints, this still results in a value function that is convex (with respect to the state variables). (An alternative approach to expanding the state-space is to use a noise-dependent term in the approximation of the value function [2].) However, if the noise is in the objective, then additional state variables that keep track of this noise will show up in the objective function of the next stage; this leads to a concave value function with respect to these state variables. As a result, the value function is a saddle function. For this reason, standard cutting plane algorithms cannot give valid lower bounds, precluding the modelling of stagewise dependence within the objective coefficients.

Typically, this is partially overcome by modelling the dependent process as a Markov chain with a set of discrete states [3; 4; 5]. However, this method can become computationally expensive since cutting plane lower bounds cannot be shared and must be added for all of the discretized Markov states. Moreover, if the process is multi-dimensional, the number of Markov states required to give a sufficient level of accuracy becomes prohibitive. In stochastic dynamic programming (SDP), a common approach to overcome this restriction is to use an interpolation scheme to evaluate the value function at points in the state space that are not in the discretized set. In the SDDP setting, this corresponds to interpolating between the value functions of different Markov states [6; 7].

The work of Baucke et al. [8] presents a new method for solving multistage minimax stochastic programming problems; this work details a procedure for the generation of valid lower (and upper) bound functions for a saddle function. Our method incorporates these bounding functions and is presented in a style of a standard SDDP implementation, enabling the solution of multistage stochastic programming problems with stagewise-dependent uncertainty in the objective function.

This paper is laid out as follows. In Section 2 we provide a formulation for our multistage stochastic program, and show that the cost-to-go is a saddle function. Then in Section 3 we present the method for generating valid lower representations for saddle functions. In Section 4 we propose an algorithm to solve such problems and provide a convergence proof. Next, in Section 5 we present an example with a discussion on the computational performance of the algorithm. Finally in Section 6 we provide concluding remarks.

2. Formulation

In this work, we consider multistage stochastic linear programming problems with uncertainty in both the right-hand sides and the objective function. We follow the terminology of Philpott

[*]Corresponding author
Email address: `a.downward@auckland.ac.nz` (Anthony Downward)
[1]Private Bag 92019, Auckland, 1142, New Zealand

and Guan in [9], restricting our attention to multistage stochastic programs with the following properties:

(1) The set of random noise outcomes Ω_t in each stage $t = 2, 3, \ldots, T$ is discrete and finite: $\Omega_t = \{1, \ldots, N_t\}$, with a known probability p_t^ω for each outcome $\omega \in \Omega_t$.

(2) The random noise realizations in each stage are independent of any previous realizations.

(3) The optimization problem in each stage has at least one optimal solution for all reachable states.

Note that we also follow their convention of including the control variables in the state space for notational convenience (i.e., the state space is extended to incorporate both states and controls). Moreover, we relax the assumption (A1) in [9], thereby allowing random quantities to appear in any part of the optimization problem (not just the right-hand sides of the constraints).

Our optimization problem, henceforth **SP**, seeks a policy (this is essentially a mapping from a state and noise outcome, at each time period, to a control) that minimizes the expected cost over the time horizon. In order to present this problem mathematically, we will define a *scenario*, s, to be a sequence of noise outcomes for stages $t \in \{1, \ldots, T\}$, with the set $\mathcal{Z}$ containing all such scenarios with positive probability. Moreover, for a given scenario $s \in \mathcal{Z}$, we denote the noise observation at the start of stage t by ω_t^s. Further, in order to ensure that the model is non-anticipative, we define $\mathcal{H}_t$ to be the set of all possible distinct noise realization sequences from stage 1 to stage t. Moreover, we set $\mathcal{H} = \bigcup_{t=1}^T \mathcal{H}_t$. For each $h \in \mathcal{H}$ we define $\mathcal{Z}^h \subseteq \mathcal{Z}$ to be the set of scenarios that start with noise realization sequence h.

$$\begin{array}{lll}\mathbf{SP}: & \min\limits_{x_t(s), \hat{x}_t^h} & \mathbb{E}_{s\in\mathcal{Z}}\left[\sum_{t=1}^T y_t(s)^\top Q_t x_t(s)\right] \\ & \text{subject to} & A_t^{\omega_t^s} x_t(s) + a_t^{\omega_t^s} \geq x_{t-1}(s),\ \forall t \in \{1, \ldots, T\},\ \forall s \in \mathcal{Z} \\ & & x_t(s) = \hat{x}_t^h,\ \forall t \in \{1, \ldots, |h|\},\ \forall s \in \mathcal{Z}^h,\ \forall h \in \mathcal{H},\end{array}$$

where $x_t(s)$ is a vector comprising the state of the system at the end of stage t, as well as the control action made during the stage, in scenario s, and $\hat{x}_t^h$ is a variable used to enforce the non-anticipativity constraint, ensuring that all scenarios with the same sequence of noise realizations up to stage $|h|$ have the same state. For scenario s in stage t, the objective function coefficient vector $y_t(s)$ follows a stochastic process of the form:

$$y_t(s) = B_t^{\omega_t^s} y_{t-1}(s) + b_t^{\omega_t^s},\ \forall t \in \{1, \ldots, T\},\ \forall s \in \mathcal{Z},$$

which is independent of our control actions. Note that Q_t is a matrix, $A_t^{\omega_t^s}$ and $B_t^{\omega_t^s}$ are square matrices, and $a_t^{\omega_t^s}$, $b_t^{\omega_t^s}$ are vectors, which may vary with the stage t and scenario s. We also have initial states $x_0(s) = x_0$, $y_0(s) = y_0$, and noise outcomes $\omega_1^s = \omega_1$, which are known and shared by all scenarios.

Typically SDDP would not be able to be applied to this type of problem since the stochastic process $\{y_t\}$ applies to the objective coefficients, which leads to a non-convex cost-to-go function. Due to this, the stochastic process would usually be approximated by modelling it as a discrete Markov chain. SDDP could be applied by generating separate cuts for each Markov state, at each stage of the problem. We present an alternative approach in this paper, which needs no such approximation.

SP can be decomposed (using a nested Benders approach) into a series of stages, each written in the following form:

$$\begin{array}{rl}\mathbf{SP}_t:\ V_t\left(x_{t-1}, y_{t-1}, \omega_t\right) = \min\limits_{x_t} & y_t^\top Q_t x_t + \ldots \\ & \ldots \mathbb{E}_{\omega_{t+1}\in\Omega_{t+1}}\left[V_{t+1}\left(x_t, y_t, \omega_{t+1}\right)\right] \\ \text{subject to} & A_t^{\omega_t} x_t + a_t^{\omega_t} \geq x_{t-1},\end{array}$$

with

$$y_t = B_t^{\omega_t} y_{t-1} + b_t^{\omega_t}.$$

y_t is a new type of state vector (an *objective-state*), which appears in the objective function, and evolves independently of any control actions taken. This objective-state enables the modelling of stagewise dependent objective uncertainty. On the other hand, x_t is a standard SDDP-type state/control vector appearing in the right-hand side of the constraints. Specifically, x_{t-1} and y_{t-1} are vectors comprising the system state at the beginning of period t, and ω_t is the noise outcome observed prior to taking any control action in period t.

Given initial states, x_0 and y_0, as well as the noise in stage 1, ω_1, we wish to find the optimal policy that minimizes the expected total cost $V_1(x_0, y_0, \omega_1)$. Note that, in the final time period, we assume (without loss of generality) that the cost-to-go is set to 0, i.e. $V_{T+1}(\cdot, \cdot, \cdot) \equiv 0$.

We define the expected cost-to-go function at the end of time period t (prior to observing the noise in period $t+1$) as:

$$\mathcal{V}_{t+1}(x_t, y_t) = \mathbb{E}_{\omega\in\Omega_{t+1}}\left[V_{t+1}\left(x_t, y_t, \omega\right)\right]. \tag{1}$$

Unlike in standard SDDP decompositions, $\mathcal{V}_{t+1}(x_t, y_t)$ is not a convex function. Specifically, the following lemma is presented in order to show that $\mathcal{V}_{t+1}(x_t, y_t)$ is a saddle function that is convex in x_t and concave in y_t, for all $t \in \{1, \ldots, T-1\}$.

Lemma 1. *For all $t \in \{1, \ldots, T\}$, the function $\mathcal{V}_t(x_{t-1}, y_{t-1})$, defined by equation (1) and the optimization problem* $\mathbf{SP}_t$*, is a saddle function that is convex with respect to x_{t-1} and concave with respect to y_{t-1}.*

Proof. We will prove this by induction. First suppose that $\mathcal{V}_{t+1}(x_t, y_t)$ is a saddle function, which is convex in x_t and concave in y_t; we know this is true for $t = T$, since $\mathcal{V}_{T+1}(x_T, y_T) = 0$.

Let us consider the convexity of $V_t(x_{t-1}, y_{t-1}, \omega_t)$ with respect to x_{t-1}, for a fixed y_{t-1} and ω_t. If y_{t-1} and ω_t are fixed, so too is y_t; this reduces $\mathbf{SP}_t$ to a convex optimization problem with x_{t-1} appearing linearly on the right-hand side of the set of constraints. Therefore, we know that $V_t(x_{t-1}, y_{t-1}, \omega_t)$ is a convex function of x_{t-1}.

Now let us consider the concavity of $V_t(x_{t-1}, y_{t-1}, \omega_t)$ with respect to y_{t-1}, for a fixed x_{t-1} and ω_t. From the definition of $\mathbf{SP}_t$, we have

$$\begin{aligned} V_t\left(x_{t-1}, y_{t-1}, \omega_t\right) = \min_{x_t} \quad & (B_t^{\omega_t} y_{t-1} + b_t^{\omega_t})^\top Q_t x_t + \dots \\ & \dots \mathcal{V}_{t+1}(x_t, B_t^{\omega_t} y_{t-1} + b_t^{\omega_t}) \\ \text{subject to} \quad & A_t^{\omega_t} x_t + a^{\omega_t} \geq x_{t-1}, \end{aligned}$$

which has an objective function that is concave in y_{t-1}. From Theorem 5.5 of [10] we know that, since $V_t(x_{t-1}, y_{t-1}, \omega_t)$ is the pointwise minimum (keeping x_{t-1} and ω_t, fixed) of a set of concave functions, it is also concave in y_{t-1}.

From the definition in (1) and using induction one can see that $\mathcal{V}_t(x_{t-1}, y_{t-1})$ is a saddle function, which is convex in x_{t-1} and concave in y_{t-1}, for $t \in \{1, \dots, T\}$. □

Lemma 1 gives the result that our cost-to-go function is a saddle function. This is different from standard decompositions of multistage stochastic programs, where the cost-to-go is convex, allowing a lower bound function to be approximated by linear cutting planes. Instead we need to approximate the expected cost-to-go saddle function $\mathcal{V}_{t+1}(x_t, y_t)$ through lower bounding functions, which we term *saddle-cuts*. These saddle-cuts were first introduced in [8], in the context of stochastic minimax dynamic programs. In the next section we outline our implementation of these saddle-cuts to provide lower bound approximations for our cost-to-go saddle function.

3. Saddle function lower bounds

This section provides a derivation of the saddle-cuts of [8] for our context. In particular, we will show how these cuts can be implemented into an optimization problem to yield valid lower bounds to a saddle function.

Consider a real-valued saddle function $\mathcal{S}(x, y)$ defined on the convex and compact set $\mathcal{X} \times \mathcal{Y}$; in particular, $\mathcal{S}(x, y)$ is convex in x, and both concave and Lipschitz continuous in y. Moreover, let us specifically assume that the L_∞-norms of the supergradients of $\mathcal{S}(x, y)$, with respect to y, are bounded from above by some Lipschitz constant ν.

Now consider a finite set of points $\mathcal{P} \subset \mathcal{X} \times \mathcal{Y}$, and suppose that for each point $(x_p, y_p) \in \mathcal{P}$ there is some corresponding saddle function $\mathcal{S}_p(x, y)$ defined over $\mathcal{X} \times \mathcal{Y}$ (which is also convex in x and concave in y) such that $\mathcal{S}_p(x, y) \leq \mathcal{S}(x, y)$, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. For each such saddle function $\mathcal{S}_p(x, y)$, we sample one point (x_p, y_p), acquiring the value of this function, $\theta_p = \mathcal{S}_p(x_p, y_p)$, and a local subgradient vector of the function, with respect to x, π_p.

In order to construct a lower bound for $\mathcal{S}(x, y)$, let us define the following optimization problem:

$$\begin{aligned} \mathcal{R}_\mathcal{P}(x, y) = \min_{\mu, \varphi} \quad & y^\top \mu + \varphi \\ \text{subject to} \quad & y_p^\top \mu + \varphi \geq \theta_p + \pi_p^\top (x - x_p), \quad \forall (x_p, y_p) \in \mathcal{P} \\ & \|\mu\|_\infty \leq \nu. \end{aligned} \tag{2}$$

Since $\mathcal{R}_\mathcal{P}(x, y)$ is the optimal value function of a linear program (where x appears in the right-hand side of the constraints and y is vector of objective coefficients), it is a piecewise saddle function. Moreover, ν provides a bound on the supergradients of $\mathcal{R}_\mathcal{P}(x, y)$ with respect to y. Theorem 1, below, gives the result that $\mathcal{R}_\mathcal{P}(x, y)$ is a lower bound for $\mathcal{S}(x, y)$ on $\mathcal{X} \times \mathcal{Y}$. However, we will first prove Lemma 2, which states that it is a lower bound at each of the sample points.

Lemma 2. *$\mathcal{S}_q(x_q, y_q) \leq \mathcal{R}_\mathcal{P}(x_q, y_q) \leq \mathcal{S}(x_q, y_q)$, for all $(x_q, y_q) \in \mathcal{P}$, for any finite set of sample points $\mathcal{P} \subset \mathcal{X} \times \mathcal{Y}$.*

Proof. By substituting any sample point $(x_q, y_q) \in \mathcal{P}$ into (2), we know from the first constraint that $\mathcal{R}_\mathcal{P}(x_q, y_q) \geq \theta_q = \mathcal{S}_q(x_q, y_q)$, ensuring that the problem is bounded. We will proceed to prove that there exists a μ^* yielding a feasible solution $(\mu, \varphi) = (\mu^*, \mathcal{S}(x_q, y_q) - y_q^\top \mu^*)$, with an objective function value of $\mathcal{S}(x_q, y_q)$.

Suppose we set μ^* to be a supergradient of $\mathcal{S}(x, y)$ with respect to y at (x_q, y_q). Our earlier assumption that the L_∞-norm of the supergradients of $\mathcal{S}(x, y)$ are less than or equal to ν ensures that the second constraint of (2) is satisfied. Now since $\mathcal{S}(x, y)$ is concave with respect to y, we know that:

$$(y_p - y_q)^\top \mu^* + \mathcal{S}(x_q, y_q) \geq \mathcal{S}(x_q, y_p), \quad \forall (x_p, y_p) \in \mathcal{P}. \tag{3}$$

Moreover, since $\mathcal{S}_p(x, y)$ is convex with respect to x, $\theta_p = \mathcal{S}_p(x_p, y_p)$, and π_p is a local subgradient, we know that:

$$\mathcal{S}_p(x_q, y_p) \geq \theta_p + \pi_p^\top (x_q - x_p), \quad \forall (x_p, y_p) \in \mathcal{P}. \tag{4}$$

Since $\mathcal{S}(x_q, y_p) \geq \mathcal{S}_p(x_q, y_p)$, equations (3) and (4) give

$$(y_p - y_q)^\top \mu^* + \mathcal{S}(x_q, y_p) \geq \theta_p + \pi_p^\top (x_q - x_p), \quad \forall (x_p, y_p) \in \mathcal{P},$$

which means that our solution $(\mu, \varphi) = (\mu^*, \mathcal{S}(x_q, y_q) - y_q^\top \mu^*)$ also satisfies the first constraint of (2), and therefore is feasible, implying that $\mathcal{R}_\mathcal{P}(x_q, y_q) \leq \mathcal{S}(x_q, y_q)$. □

Corollary 1. *Suppose that some point $(x_q, y_q) \in \mathcal{P}$ corresponds to a saddle function $\mathcal{S}_q(x, y) = \mathcal{S}(x, y)$, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. This implies that $\mathcal{R}_\mathcal{P}(x_q, y_q) = \mathcal{S}(x_q, y_q)$.*

Proof. From Lemma 2 above, we have $\mathcal{S}_q(x_q, y_q) \leq \mathcal{R}_\mathcal{P}(x_q, y_q) \leq \mathcal{S}(x_q, y_q)$. Therefore, if $\mathcal{S}_q(x_q, y_q) = \mathcal{S}(x_q, y_q)$ this implies that $\mathcal{R}_\mathcal{P}(x_q, y_q) = \mathcal{S}(x_q, y_q)$. □

Theorem 1. *$\mathcal{R}_\mathcal{P}(x, y) \leq \mathcal{S}(x, y)$ for any finite set of sample points $\mathcal{P} \subset \mathcal{X} \times \mathcal{Y}$, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

Proof. For the sake of contradiction, suppose that there exists a point $(\tilde{x}, \tilde{y}) \in \mathcal{X} \times \mathcal{Y}$ for which $\mathcal{R}_\mathcal{P}(\tilde{x}, \tilde{y}) > \mathcal{S}(\tilde{x}, \tilde{y})$. By defining $\mathcal{P}^* = \mathcal{P} \cup (\tilde{x}, \tilde{y})$, by Lemma 2, we have $\mathcal{R}_{\mathcal{P}^*}(\tilde{x}, \tilde{y}) \leq \mathcal{S}(\tilde{x}, \tilde{y})$. This implies that $\mathcal{R}_{\mathcal{P}^*}(\tilde{x}, \tilde{y}) < \mathcal{R}_\mathcal{P}(\tilde{x}, \tilde{y})$, which is a contradiction since the feasible region of $\mathcal{R}_{\mathcal{P}^*}$ is a subset of the feasible region of $\mathcal{R}_\mathcal{P}$, implying that $\mathcal{R}_{\mathcal{P}^*}(\tilde{x}, \tilde{y}) \geq \mathcal{R}_\mathcal{P}(\tilde{x}, \tilde{y})$. Thus $\mathcal{R}_\mathcal{P}(x, y) \leq \mathcal{S}(x, y)$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$, as required. □

In order to apply this bounding function concept to $\mathcal{V}_{t+1}(x_t, y_t)$, defined in (1), first observe that the set $\mathcal{X} \times \mathcal{Y}$ can be defined to be the convex hull of the set of reachable states (x_t, y_t) for the given stage. Secondly, the value functions $\mathcal{V}_{t+1}(x_t, y_t)$ defined in $\mathbf{SP}_t$, do indeed exhibit the property of bounded supergradients in y_t required for the bounding function $\mathcal{R}_{\mathcal{P}}(x, y)$ to be valid. This property arises from the following: $\mathbf{SP}_t$ is always feasible (admitting finitely bounded duals), cost coefficient matrices are bounded and finite, and finally that there are a finite number of stages. However, although the supergradients are bounded, they may not be known. Given that Lemma 2 (and hence Theorem 1) relies on ν exceeding the L_∞-norm of the largest supergradient, it is necessary to estimate this. In Section 5.2, we examine the trade-offs associated with the choice of ν, computationally.

4. Proposed algorithm and convergence

Baucke et al. in [8] provide an ϵ-convergence result for their algorithm which solves a more general class of problems, with minimax saddle function stage problems. This requires the definition of an upper bound function as well as a lower bound. In our setting, since y_t is exogenously determined we can forgo the computation of an upper bound function (and hence forgo deterministic convergence). Instead we prove the almost sure convergence of a different algorithm. Moreover, the formulation that we present here is more readily implementable in existing SDDP libraries (e.g. SDDP.jl [11]).

By Lemma 1, the cost-to-go function $\mathcal{V}_{t+1}(x_t, y_t)$, defined in equation (1), is convex with respect to x_t, and concave with respect to y_t. Therefore, we can form a lower bound on the cost-to-go using the saddle-cuts from Section 3. However, since we have a stochastic program we form saddle-cuts for $\mathcal{V}_{t+1}(x_t, y_t)$ by taking an expectation with respect to the set of scenarios in the next stage. Moreover, since we consider a multistage problem, the lower bound saddle-cuts in stage t will be formed based on the approximate saddle function for stage $t+1$. Saddle-cuts are added in each iteration, continually improving the lower bound. This concept extends naturally from SDDP, which uses nested Benders cuts.

As with SDDP-type methods, we define an approximation of problem $\mathbf{SP}_t$. In iteration k, this approximation, called $\mathbf{AP}_t^k$, replaces the expected cost-to-go term with a set of saddle-cut lower-bounds generated in iterations 1 through $k-1$.

$\mathbf{AP}_t^k(x_{t-1}^k, y_{t-1}^k, \omega_t)$:

$$\begin{aligned}
\theta_t^k(x_{t-1}^k, y_{t-1}^k, \omega_t) = \min_{x_t, \mu_t, \varphi_t} \quad & y_t^{k\top} Q_t x_t + y_t^{k\top}\mu_t + \varphi_t \\
\text{subject to} \quad & A_t^{\omega_t} x_t + a_t^{\omega_t} \geq x_{t-1}^k & [\pi_t^k] \\
& y_t^{j\top}\mu_t + \varphi_t \geq \alpha_{t+1}^j + \beta_{t+1}^{j\top} x_t, \\
& \qquad j = 1, 2, \ldots, k-1 & [\rho_t^{jk}] \\
& \varphi_t \geq \sigma \\
& \|\mu_t\|_\infty \leq \nu,
\end{aligned}$$

where

$$y_t^k = B_t^{\omega_t} y_{t-1}^k + b_t^{\omega_t}.$$

α_{t+1}^j, β_{t+1}^j are parameters of the *expected* saddle-cut added in iteration j, which are piecewise constant functions of x_t^j (see the description of Algorithm 1 for a precise definition); and π_t^k, ρ_t^{jk} are dual vectors, corresponding to the first and second constraints, respectively. Recall that for a fixed ω_t, $\theta_t^k(x_{t-1}^k, y_{t-1}^k, \omega_t)$ is a saddle function, and furthermore, in the final stage T the cost-to-go is 0, so we can remove the μ_t and φ_t variables. Algorithm 1 constructs saddle-cuts for $\mathbf{AP}_t^k$ that successively improve the approximation of the true expected cost-to-go saddle functions. In order to avoid an unbounded problems in iteration 1, the parameter σ provides a lower bound for φ_t, and must be set to some sufficiently small value so that it gives a valid lower bound for the expected cost-to-go.

In each iteration $k \in \{1, 2, \ldots\}$ of Algorithm 1, we select a scenario $s_k \in \mathcal{Z}$, then we compute a trajectory of feasible solutions $\left\{(x_t^k, y_t^k) : t \in \{1, \ldots, T-1\}\right\}$ on each forward pass, and then add a single saddle-cut (lower bound) for $\mathbf{AP}_t^k$ for each stage $t \in \{1, \ldots, T-1\}$ on each backward pass.

A key distinction between our problem and ones that can be solved using SDDP-type algorithms, as examined in [9], is that due to the random noise affecting the objective function, in order to construct a valid saddle-cut, every realization of the next stage's random noise must be sampled at each stage on the backward pass. However, just as in SDDP-type algorithms, a statistical estimate of an upper bound can be obtained by a Monte Carlo simulation of the policy (see, e.g., [1]).

The description of Algorithm 1 details the specific procedure to solve $\mathbf{SP}$, for given starting states (x_0, y_0), and noise observation ω_1. In particular, we consider two variants of Algorithm 1, as defined below.

(a) For all $k \geq 1$ we set s_k to be a repeating (ordered) infinite sequence of all the scenarios in $\mathcal{Z}$.
(b) For each $k \geq 1$ we let s_k be randomly and independently sampled from the set of scenarios $\mathcal{Z}$; note that the probability of selecting a particular scenario need not match its true probability, however, it must be non-zero. This also yields an infinite sequence of scenarios.

Since the first noise outcome is deterministic, for each scenario $s \in \mathcal{Z}$, $\omega_1^s = \omega_1$ (all other outcomes in stage 1 have zero probability, so the corresponding scenarios are not in $\mathcal{Z}$).

In what follows, we show that Algorithm 1(a) converges to an optimal solution in a finite number of iterations, whereas Algorithm 1(b) *almost surely* converges to an optimal solution in a finite number of iterations. In order to prove this convergence of the variants of Algorithm 1, we will first show that there are only a finite number of possible saddle-cuts that can be constructed. In order to prove this, we cannot *directly* invoke Lemma 1 from [9], since we have different stage problems. Therefore we present Lemma 3 below to show that the collection of distinct saddle-cuts, defined by parameters $(y_t^k, \alpha_{t+1}^k, \beta_{t+1}^k)$, is nevertheless provably finite. This Lemma follows the same

Algorithm 1: Stochastic dual dynamic programming with objective-states

```
set $\nu$ to some constant known to exceed the Lipschitz
  constants of $\mathcal{V}_t(x_{t-1}, y_{t-1}),\ \forall t \in \{1,\ldots,T\}$,
set $\sigma$ to some constant known to be less than
  $\mathcal{V}_t(x_{t-1}, y_{t-1}),\ \forall t \in \{1,\ldots,T\}$,
set $k = 1$
while not converged do
    set $x_0^k = x_0,\ y_0^k = y_0$
    /* Forward Pass                                   */
    for $t = 1 : T-1$ do
        solve $\mathbf{AP}_t^k(x_{t-1}^k, y_{t-1}^k, \omega_t^{s_k})$ (returning an extreme
          point solution)
        set $x_t^k$ to the value of $x_t$ in the optimal solution
    end
    /* Backward Pass                                  */
    for $t = T-1 : 1$ do
        for $\omega \in \Omega_{t+1}$ do
            solve $\mathbf{AP}_{t+1}^k(x_t^k, y_t^k, \omega)$ (returning an extreme
              point dual solution)
            set $\theta_{t+1}^{k,\omega}$ to the optimal objective value $\theta_{t+1}^k$
            set $\pi_{t+1}^{k,\omega}$ to the value of $\pi_{t+1}^k$ in the optimal
              dual solution
        end
        set $\beta_{t+1}^k = \mathbb{E}_{\omega\in\Omega_{t+1}}\left[\pi_{t+1}^{k,\omega}\right]$
        set $\alpha_{t+1}^k = \mathbb{E}_{\omega\in\Omega_{t+1}}\left[\theta_{t+1}^{k,\omega}\right] - \beta_{t+1}^{k\top} x_t^k$
        add the saddle-cut $y_t^{k\top}\mu_t + \varphi_t \geq \alpha_{t+1}^k + \beta_{t+1}^{k\top} x_t$ to
          $\mathbf{AP}_t^k$
    end
    solve $\mathbf{AP}_1^k(x_0, y_0, \omega_1)$ to obtain $\theta_1^k$ as a lower bound
    increment $k$ to $k+1$
end
```

inductive reasoning as Lemma 1 of [9], with some of their results applied (without derivation) in our proof.

Lemma 3. *In iteration k of Algorithm 1, for each $t \in \{1, 2, \ldots, T-1\}$, define the collection of saddle-cuts by the set*

$$\mathcal{G}_t^k = \left\{\left(y_t^j, \alpha_{t+1}^j, \beta_{t+1}^j\right) : j = 1, 2, \ldots, k-1\right\}.$$

Then for any sequence $\mathcal{G}_t^k, k \in \{1, 2, \ldots\}$ generated by Algorithm 1, there exists m_t such that for all k: $\left|\mathcal{G}_t^k\right| \leq m_t$. Furthermore, there exists k_t, so that if $k \geq k_t$ then $\mathcal{G}_t^k = \mathcal{G}_t^{k_t}$.

Proof. This lemma arises from the finite number of possible realizations for the y_t vector and the finite number of extreme points of the feasible region in each stage problem. Since, in the final stage, $\mathbf{AP}_T^k$ is a linear program with a finite number of variables and constraints, by Assumption 1 the values of α_T and β_T in the saddle-cuts for stage $T-1$ will correspond to an expectation over extreme points of the feasible region of the dual problem; thus there will be a finite set of possible values. Moreover, there is a finite number of values that y_{T-1} can take. This gives a finite bound, m_{T-1}, on the number of saddle-cuts, i.e. $\left|\mathcal{G}_{T-1}^k\right| \leq m_{T-1}$. Due to there being a finite number of unique saddle-cuts, there will exist some iteration k_{T-1} such that for $k \geq k_{T-1}$, $\mathcal{G}_{T-1}^k = \mathcal{G}_{T-1}^{k_{T-1}}$; moreover, $\mathbf{AP}_{T-1}^{k_{T-1}}$ will also be a linear program with a finite number of variables and constraints. This can be applied inductively to give the result, in a similar way as is presented in Lemma 1 of [9]. □

Using the result of Lemma 3 (that there are a finite number of saddle-cuts possible at each stage) we will now show that Algorithm 1(a) is convergent.

Lemma 4. *Algorithm 1(a), converges after a finite number of iterations to an optimal policy for* **SP**.

Proof. By Lemma 3, for each $t \in \{2, \ldots, T\}$, there exists k_t, such that if $k > k_t$ then $\mathcal{G}_t^k = \mathcal{G}_t^{k_t}$. This means that there will be no further change in the cuts defining $\mathcal{V}_t\left(x_{t-1}, y_{t-1}\right)$. We define $\bar{k} = \max_t\{k_t\}$, to be the minimum number of iterations before all stages of the model no longer change.

Observe that since the stochastic process for y_t is discrete, there are a finite number of possible values for y_t for each $t \in \{2, \ldots, T\}$. Moreover, since $\mathcal{G}_t^k$ is no longer changing for $k > \bar{k}$, we arrive at a fixed trajectory of states $x_t(s)$ for each scenario s. We define the values of $y_t(s)$ in a similar way (however, these are explicitly determined from the random noise outcomes for scenario s). We will now show that the states are optimal for the problem $\mathbf{SP}_t$ for all stages and for each scenario.

First note that the optimal value function for $\mathbf{AP}_T^{\bar{k}}$ is the same as that for $\mathbf{SP}_T$; this implies that

$$\mathbb{E}_{\omega\in\Omega_T}\left[\theta_T^{\bar{k}}\left(x_{T-1}(s), y_{T-1}(s), \omega\right)\right] = \mathcal{V}_T\left(x_{T-1}(s), y_{T-1}(s)\right),\ \forall s \in \mathcal{Z}. \tag{5}$$

We now use induction to prove that at iteration $\bar{k}$ this is true for all $t \in \{1, \ldots, T\}$. Suppose the analogous result to (5) is true for stage $t+1$, we will show that is it also true for stage t; that is:

$$\mathbb{E}_{\omega\in\Omega_t}\left[\theta_t^{\bar{k}}\left(x_{t-1}(s), y_{t-1}(s), \omega\right)\right] = \mathcal{V}_t\left(x_{t-1}(s), y_{t-1}(s)\right),\ \forall s \in \mathcal{Z}. \tag{6}$$

We will prove this by contradiction. First, we note that by Theorem 1 we are adding saddle-cuts that are guaranteed to not exceed the true cost-to-go, so in order for (6) to not hold, there must exist some scenario $\hat{s}$ such that

$$\theta_t^{\bar{k}}\left(x_{t-1}(\hat{s}), y_{t-1}(\hat{s}), \omega_t^{\hat{s}}\right) < V_t\left(x_{t-1}(\hat{s}), y_{t-1}(\hat{s}), \omega_t^{\hat{s}}\right).$$

However from Algorithm 1(a), we have

$$\begin{aligned}
\theta_t^{\bar{k}}\left(x_{t-1}(\hat{s}), y_{t-1}(\hat{s}), \hat{\omega}\right) = \min_{x_t, \mu_t, \varphi_t} \quad & y_t^\top Q x_t + y_t^\top \mu_t + \varphi_t \\
\text{subject to} \quad & A_t^{\hat{\omega}} x_t + a_t^{\hat{\omega}} \geq x_{t-1}(\hat{s}) \\
& y_t^{j\top}\mu_t + \varphi_t \geq \alpha_{t+1}^j + \beta_{t+1}^{j\top} x_t \\
& \qquad j = 1, 2, \ldots, \bar{k}-1 \\
& \varphi_t \geq \sigma \\
& \|\mu_t\|_\infty \leq \nu,
\end{aligned}$$

where $y_t = B_t^{\hat{\omega}} y_{t-1}(\hat{s}) + b_t^{\hat{\omega}}$, and for notational convenience we set $\hat{\omega} = \omega_t^{\hat{s}}$. Furthermore, we denote the optimal solution by $(x_t^*, \mu_t^*, \varphi_t^*)$, where $x_t^* = x_t(\hat{s})$.

If $y_t^\top \mu_t^* + \varphi_t^* < \mathcal{V}_{t+1}(x_t^*, y_t)$, from Corollary 1 (given our chosen value for ν) we know that this means that there exists some random noise outcome, which we have not sampled in the backward pass, since if we had sampled it then we would have added a cut that would force $y_t^\top \mu_t^* + \varphi_t^* = \mathcal{V}_{t+1}(x_t^*, y_t)$. However, this contradicts the backward pass of Algorithm 1(a), which requires us to sample all random noise outcomes for the next stage to generate the expected cut. Given this contradiction and due to the enumeration of scenarios in the forward pass of Algorithm 1(a), equation (6) must hold. Therefore, after a finite number of iterations $\bar{k}$, Algorithm 1(a) converges to the optimal policy for **SP**. □

Although we have shown that Algorithm 1(a) converges, it is computationally intractable, since it enumerates all of the scenarios. In Theorem 2 we prove almost sure convergence of Algorithm 1(b) in a finite number of iterations, but first we present a short corollary.

Corollary 2. *Suppose, in each stage $t \in \{1, \ldots, T-1\}$, there exists in $\mathbf{AP}_t^1$ some initial (finite) set of saddle-cuts that form a valid lower bound for the true cost-to-go function $\mathcal{V}_{t+1}(x_t, y_t)$. Then Lemma 4 still holds if we begin Algorithm 1(a) from this point.*

Proof. Introducing these additional valid saddle-cuts to $\mathbf{AP}_t^1$ has no bearing on the application of Theorem 1 and Lemma 3 in the convergence proof of Lemma 4. Although the set of saddle-cuts produced may be different, there are still a finite number. □

Theorem 2. *Algorithm 1(b) converges with probability 1 to an optimal policy for* **SP** *in a finite number of iterations.*

Proof. Due to the forward pass sampling method given in Algorithm 1(b), without the stopping criteria, we know, by the second Borel-Cantelli lemma [12], that after some iterations we would almost surely realize a finite sequence of iterations that is identical to those traversed using Algorithm 1(a) until convergence. This gives us an initial set of cuts before the iterations of Algorithm 1(a), and thus from Corollary 2, we know that Algorithm 1(b) will converge in a finite number of iterations to the optimal policy for **SP** with probability 1. □

5. Computational Experiments

We implement the hydro-bidding problem with price uncertainty example from Chapter 6.1 of [7]. The example has two reservoirs in a cascade over a planning horizon of 12 periods (i.e. $T = 12$). On the outflow of each reservoir is a turbine followed by a river that leads to the next reservoir in the cascade. The last reservoir discharges into the sea. The goal of the agent is to choose the quantity of water to flow through each turbine (and thereby, produce electricity), as well as the quantity of water to spill over the top of each reservoir, in order to maximize the revenue gathered from selling the electricity generated on the spot market. In each stage t, we denote the combination of spills and flows as u_t. We denote the feasibility constraints on the action of the agent (i.e. maximum and minimum flow limits) as $u_t \in \mathcal{U}$. Based on the action u_t chosen by the agent, the state of the system x_t (the quantity of water in each of the reservoirs at the start of stage t) will transition according to the linear function $x_t = F(x_{t-1}, u_t)$. In addition, the agent earns an immediate reward by selling $G(u_t)$ units of power at the current spot-price y_t. They may also incur a penalty of $C(u_t)$ if they exceed some operating limits of the reservoir (i.e. a cost of spillage). Unless explicitly stated, all parameters are identical to the description given in [7].

5.1. First-order auto-regressive price process

We first consider a spot-price process y_t which is modelled as an auto-regressive lag 1 process: $y_t = 0.5y_{t-1} + 0.5b(t) + \omega_t$, where $\Omega_t = \{-4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5\}$, and $b(t)$ is a deterministic drift term of the auto-regressive process as described in [7]. Each of the realizations ω_t in Ω_t is sampled with uniform probability. This hydro-bidding model can be described by the stage problem:

$$\begin{aligned}
\mathbf{SP}_t: \; V_t(x_{t-1}, y_{t-1}, \omega_t) = \min_{u_t, x_t} \quad & C(u_t) - y_t G(u_t) + \ldots \\
& \ldots \mathbb{E}_{\omega_{t+1} \in \Omega_{t+1}} \left[V_{t+1}(x_t, y_t, \omega_{t+1}) \right] \\
\text{subject to} \quad & x_t = F(x_{t-1}, u_t) \\
& u_t \in \mathcal{U},
\end{aligned}$$

where $y_t = 0.5y_{t-1} + 0.5b(t) + \omega_t$.

This model was implemented in the SDDP.jl package [11] in the Julia language [13]. The model was solved using Algorithm 1(b) for 2000 iterations. (In the style of DOASA [9], each iteration of the algorithm adds one cut to each stage problem.) Every 250 iterations, we performed a Monte Carlo simulation of the policy with 250 replications in order to construct a confidence interval for the expected value of the policy. A plot of the lower and upper bounds against the number of iterations is given in Figure 1. In addition, we solved the same model using the *stochastic dynamic programming* (SDP) algorithm (implemented in the DynamicProgramming.jl package [14]). The lower bound for the problem using Algorithm 1(b) converges to the first-stage objective value (calculated using the SDP algorithm) of −\$24,855.

In this model there are two convex state variables (the quantity of water in the upstream and downstream reservoirs), and one concave objective-state variable. Thus, we can visualize the cost-to-go function by fixing one of the three state variables, and varying the other two across their domain. Figure 2 is one such visualization for the cost-to-go function at the start of stage 6 (i.e., V_6), given 50 units of water in the upstream reservoir and assuming that $\omega_6 = 0$. We can clearly see how the cost-to-go function is convex with respect to the downstream volume, and concave with respect to the spot-price.

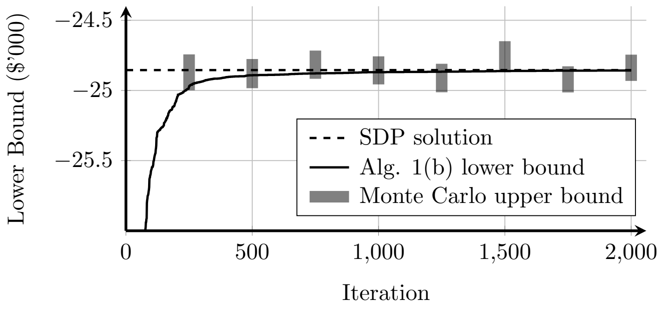

Figure 1: Convergence of the lower and upper bounds against the number of iterations.

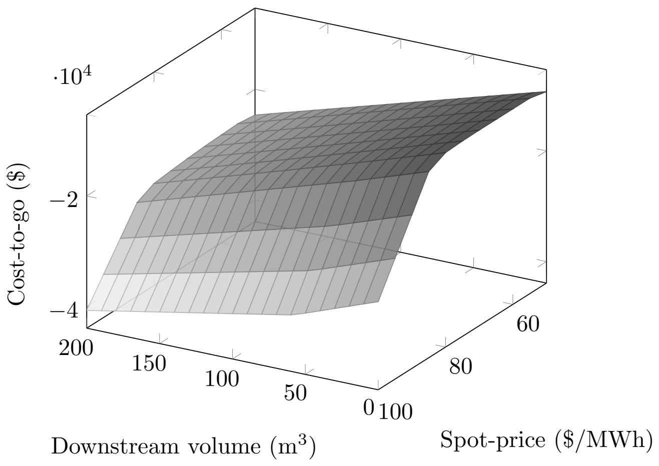

Figure 2: Visualization of the cost-to-go function at the start of stage 6.

5.2. *Choice of Lipschitz constant*

Recall that it is necessary to bound the μ_t variables in the approximate problems $\mathbf{AP}_t$ by some value ν, which we refer to as the Lipschitz constant. If ν is chosen to be too small, then the saddle cuts no longer form a valid lower bound approximation of the cost-to-go function. However, if ν is made needlessly large, then the bounds on the value functions are weakened, hampering the speed of convergence. To ensure correctness in the example above, we set a conservative value for the Lipschtiz constant of $\nu = 10^6$. To demonstrate the behavior of smaller choices for ν, in Figure 3 we plot the lower bound of the two-reservoir problem against the number of iterations for various values of ν. (Note the different number of iterations in comparison to Figure 1.) When $\nu = 0$ and $\nu = 10$, the problem converges to a sub-optimal solution. When $\nu = 100$ and $\nu = 1000$, the problem converges to the optimal solution, and the convergence rate when $\nu = 1000$ is slower than when $\nu = 100$. Choosing an appropriate value for ν in practice requires domain knowledge. In addition, we recommend solving each problem with different values of ν to see how this choice affects the lower bound and solution time.

5.3. *Second-order auto-regressive price process*

In the simple example presented in Section 5.1, the SDP approach can be faster than Algorithm 1(b) (although it depends

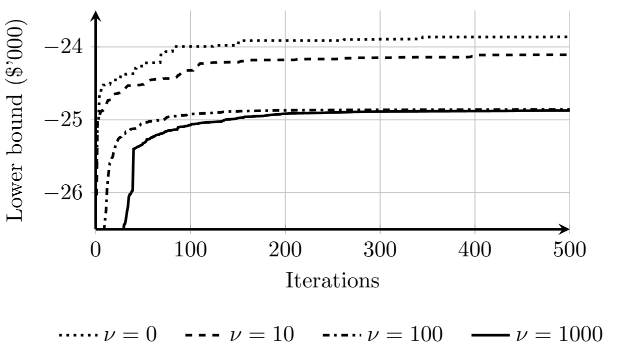

Figure 3: Convergence of the lower bound for the two-reservoir problem with varying ν.

on the level of discretization of the state and action spaces). However, it is well known that the SDP algorithm is limited by the "curse of dimensionality". To demonstrate the ability of Algorithm 1(b) to escape the curse, we extend the model to a cascade of five reservoirs, and the price process to an auto-regressive process with lag 2: $y_t = 0.5y_{t-1} + 0.5(y_{t-1} - y_{t-2}) + 0.5b(t) + \omega_t$.

There are now seven state-dimensions in the model (the five reservoir levels, plus the two prices y_{t-1} and y_{t-2}), and five action-dimensions (the quantity of water to release from each reservoir). This problem is too large to solve in a reasonable time using the SDP algorithm. However, the enlarged problem was solved using Algorithm 1(b) for 5000 iterations to form an approximately optimal policy. Then, a Monte-Carlo simulation was conducted with 1000 replications using the policy. This took approximately 2000 seconds using a single core of a Windows 7 machine with an Intel i7-4770 CPU and 16GB of memory. In Figure 4, we visualize the Monte-Carlo simulation. In each of the subplots, we plot as shaded bands in order of increasing darkness, the 0–100, 10–90, and 25–75 percentiles of the distribution of the plotted variable. The solid line corresponds to the 50th percentile. There are also two individual replications plotted: a high-price realization (thick dashed line) and low-price realization (thick dotted line).

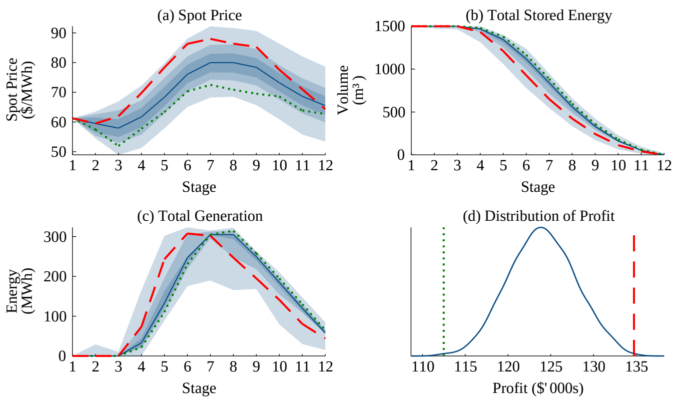

Figure 4: Monte-Carlo simulation using the optimal policy.

Over the stages 1–7, the spot-price increases (Figure 4a). Therefore, optimal policy is to conserve water in anticipation of a future higher profit. In Figure 4b, we plot the total stored energy in the system. ($1m^3$ in Reservoir 1 is worth five units as it can be used by each of the five turbines in the cascade. In contrast, $1m^3$ in Reservoir 5 is only worth one unit as it can only be used once before it flows out of the cascade.) In the low-price scenario (thick dotted line), more energy is stored in the system at a given point in time in the expectation that the spot-price will revert upwards to the mean, whereas in the high-price scenario (thick dashed line), more energy is generated (Figure 4c) in the early stages (i.e. stages 1-5) in anticipation that the futures prices will revert downwards to the mean. This leads to the distribution in profit shown in Figure 4d.

6. Conclusions

In this paper we have presented an algorithm which converges almost surely in a finite number of iteration when solving linear multistage stochastic programming problems with stagewise dependent noise in the objective function. This method enables the modelling of stochastic price processes, such as pure auto-regressive (AR) models or ones incorporating a moving average (ARMA), within the SDDP framework. These types of problems could previously only be approximated using Markov chains [3], or binary expansion techniques [15].

We have presented an example of a five-reservoir cascading river-chain with a lag 2 auto-regressive price process to show that this method is computationally tractable for moderately-sized problems (for which SDP models will run into the curse of dimensionality). We see that the converged solution provides a policy which is intuitive, yet complex, as the model is trading off profits that can be made in the current period, against future expected profits that could be made.

Furthermore, note that the algorithm is not limited to price processes, and is general to any objective-state variable, so there are many applications for this method. The key requirement is that these state variables evolve independently of the control decisions and the values of the other state variables.

This algorithm has been implemented within the SDDP.jl Julia library [11]. We provide Julia code for Algorithm 1(b) as supplementary material at `https://github.com/odow/SDDP.jl`.

Acknowledgements

The authors would like to acknowledge Prof. Andy Philpott and Assoc. Prof. Andrew Mason for useful discussions on this work, and thank the anonymous referees for their suggestions.

References

[1] M. V. F. Pereira, L. M. V. G. Pinto, Multi-stage stochastic optimization applied to energy planning, Mathematical Programming 52 (1-3) (1991) 359–375. doi:10.1007/BF01582895.

[2] G. Infanger, D. P. Morton, Cut sharing for multistage stochastic linear programs with interstage dependency, Mathematical Programming 75 (1996) 241–256.

[3] A. Gjelsvik, M. Belsnes, A. Haugstad, An algorithm for stochastic medium term hydro thermal scheduling under spot price uncertainty, in: PSCC: 13th Power Systems Computation Conference : Proceedings, Executive Board of the 13th Power Systems Computation Conference, 1999, Trondheim, 1999, p. 1328.

[4] A. B. Philpott, V. L. De Matos, Dynamic sampling algorithms for multistage stochastic programs with risk aversion, European Journal of Operational Research 218 (2) (2012) 470–483. doi:10.1016/j.ejor.2011.10.056.

[5] S. Rebennack, Combining sampling-based and scenario-based nested Benders decomposition methods: application to stochastic dual dynamic programming, Mathematical Programming 156 (1-2) (2016) 343–389.

[6] A. Gjelsvik, B. Mo, A. Haugstad, Long- and Medium-term Operations Planning and Stochastic Modelling, in: P. M. Pardalos, S. Rebennack, M. V. F. Pereira, N. A. Iliadis (Eds.), Handbook of Power Systems I, Energy Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 33–55. doi:10.1007/978-3-642-02493-1.

[7] F. Wahid, River optimization: short-term hydro-bidding under uncertainty, PhD thesis, University of Auckland, Auckland, New Zealand (2017).

[8] R. Baucke, A. Downward, G. Zakeri, A deterministic algorithm for solving multistage stochastic minimax dynamic programmes, Optimization Online.
URL http://www.optimization-online.org/DB_HTML/2018/02/6449.html

[9] A. B. Philpott, Z. Guan, On the convergence of stochastic dual dynamic programming and related methods, Operations Research Letters 36 (4) (2008) 450–455. doi:10.1016/j.orl.2008.01.013.

[10] R. Rockafellar, Convex Analysis, Princeton University Press, Princeton, New Jersey, 1992.

[11] O. Dowson, L. Kapelevich, SDDP.jl: a Julia package for Stochastic Dual Dynamic Programming, Optimization Online.
URL http://www.optimization-online.org/DB_HTML/2017/12/6388.html

[12] G. Grimmett, D. Stirzaker, Probability and Random Processes, Oxford University Press, Oxford, 1992.

[13] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A Fresh Approach to Numerical Computing, SIAM Review 59 (1) (2017) 65–98.

[14] O. Dowson, DynamicProgramming.jl: a Julia package for Stochastic Dynamic Programming, [Online; accessed 2017-10-26] (2017).
URL https://github.com/odow/DynamicProgramming.jl

[15] J. Zou, S. Ahmed, X. A. Sun, Stochastic dual dynamic integer programming, Mathematical Programming 175 (1-2) (2019) 461–502.