MIDAS: A Mixed Integer Dynamic Approximation Scheme

A.B. Philpott · F. Wahid · J.F. Bonnans

April 20, 2018

Abstract Mixed Integer Dynamic Approximation Scheme (MIDAS) is a new sampling-based algorithm for solving finite-horizon stochastic dynamic programs with monotonic Bellman functions. MIDAS approximates these value functions using step functions, leading to stage problems that are mixed integer programs. We provide a general description of MIDAS, and prove its almost-sure convergence to a $2T\varepsilon$-optimal policy for problems with T stages when the Bellman functions are known to be continuous, and the sampling process satisfies standard assumptions.

Keywords stochastic programming · approximate dynamic programming · sampling · mixed-integer programming

1 Introduction

In general, multistage stochastic programming problems are extremely difficult to solve. If one discretizes the random variable using a finite set of outcomes in each stage and represents the process as a scenario tree, then the problem size grows exponentially with the number of stages and outcomes [14]. On the other hand, if the random variables are stagewise independent then the problem can be formulated as a dynamic programming recursion, and attacked using an approximate dynamic programming algorithm.

When the Bellman functions (or value-to-go functions) are known to be convex (if minimizing) or concave (if maximizing) then they can be approximated by cutting planes. This is the basis of the popular Stochastic Dual Dynamic Programming (SDDP) algorithm originally proposed by Pereira and Pinto [12]. This creates a sequence

A.B. Philpott
Electric Power Optimization Centre
University of Auckland
Tel.: +64-9-3737599
E-mail: a.philpott@auckland.ac.nz

F. Wahid
University of Auckland and INRIA, Ecole Polytechnique, Saclay

J.F. Bonnans
INRIA, Ecole Polytechnique, Saclay

of cutting-plane outer approximations to the Bellman function at each stage by evaluating stage problems on independently sampled sequences of random outcomes. A comprehensive recent summary of this algorithm and its properties is provided by [15]. A number of algorithms that are variations of this idea have been proposed (see e.g. [4,5,10]).

The first formal proof of almost-sure convergence for SDDP-based algorithms applied to problems with polyhedral Bellman functions was provided by Philpott and Guan in [13]. Almost-sure convergence for general convex Bellman functions was recently proved using a different approach by Girardeau et al. [7], and for risk averse problems by Guigues [9].

In order to guarantee the validity of the outer approximation of the Bellman functions, SDDP-based methods require these functions to be convex (if minimizing) or concave (if maximizing). However, there are many instances of problems when the optimal value of the stage problem is not a convex (or concave) function of the state variables. For example, if the stage problem involves state variables with components appearing in both the objective function and the right-hand side of a convex optimization problem then the optimal value function might have a saddle point (unless this function is jointly convex in both the state and decision). More generally, if the stage problems are not convex then we cannot guarantee convexity of the optimal value function. This will happen when stage problems are mixed-integer programs (MIPS), or when they incorporate nonlinear effects, such as modeling head effects in hydropower production functions [3].

Our interest in problems of this type is motivated by models of hydroelectricity systems in which we seek to maximize revenue from releasing water through generating stations on a river chain. This can be modeled as a stochastic decision problem with discrete-time dynamics:

$$x_{t+1} = f_t(x_t, u_t, \xi_t),\ x_1 = \bar{x},\ t = 1, 2, \dots, T, \tag{1}$$

where f_t is an affine function. In this stochastic state equation, x_t is an element of the set of feasible states X_t, $t = 1, 2, \dots, T + 1$, the initial state $\bar{x} = x_1$ being given. For $t = 1$ to T, the ξ_t are independent random variables, elements of Ω_t, and u_t, the decision at time t, is an element of the set of feasible decisions $U_t(x_t, \xi_t)$. This means that there is a complete observation of the state, and that we are in a hazard-decision setting. Also, for each $x_t \in X_t$ and $u_t \in U_t(x_t, \xi_t)$, $f_t(x_t, u_t, \xi_t)$ is an element of X_{t+1}. The decision u_t generates a reward $r_t(x_t, u_t, \xi_t)$ in each stage; there is also a terminal reward $R(x_{T+1})$. Under technical hypotheses, the Bellman values $V_t(x)$ (maximum expected reward from the beginning of stage t onwards with state x) are well-defined and satisfy the dynamic programming principle

$$V_t(x) = \mathbb{E}_{\xi_t}\left[\sup_{u \in U_t(x, \xi_t)} \{r_t(x, u, \xi_t) + V_{t+1}(f_t(x, u, \xi_t))\}\right], \tag{2}$$

$$V_{T+1}(x) = R(x). \tag{3}$$

We are interested in the case when the X_t are subsets of $\mathbb{R}^n$, and the Bellman values are nondecreasing functions of x. For example, in a single-reservoir hydro-scheduling problem, $x = (s, p)$ might represent both the reservoir stock variable s and a price variable p, and $u = (v, l)$ represents the reservoir release v through a generator and reservoir spill l. In this case, the dynamics might be represented by

$$\begin{bmatrix} s_{t+1} \\ p_{t+1} \end{bmatrix} = \begin{bmatrix} s_t - v_t - l_t + \omega_t \\ \alpha_t p_t + (1 - \alpha_t)b_t + \eta_t \end{bmatrix},$$

where $\alpha_t \in (0, 1)$ and b_t is a time-varying average price towards which p_t is mean reverting. Here ω_t is (random) reservoir inflow, and η_t is the error term for the mean-reverting price model, so $\xi_t = [\,\omega_t\ \eta_t\,]^{\top}$. Here we might define

$$r_t(s, p, v, l, \omega_t, \eta_t) = pg(v),$$

giving the revenue earned by released energy $g(v)$ sold at price p, and

$$U_t(x_t, \xi_t) = \{(v, l) : v \in \bar{U},\ l \geq 0,\ 0 \leq -v - l + x_t + \omega_t \leq a\},$$

where a is the capacity of the reservoir, and $\bar{U}$ is the set of feasible water releases through the generating station. For such a model, it is easy to show (under the assumption that we can spill energy without penalty) that $V_t(s, p)$ is continuous and monotonic increasing in s and p. On the other hand $V_t(s, p)$ is not always concave which makes the application of standard SDDP invalid.

In this paper we propose a new extension of SDDP called Mixed Integer Dynamic Approximation Scheme (MIDAS). MIDAS uses the same algorithmic framework as SDDP but, instead of using cutting planes, MIDAS uses step functions to approximate the value function. The approximation requires an assumption that the value function is monotonic (or close to monotonic in a precise sense) in the state variables. Each approximating stage problem can be solved to global optimality as a MIP. The approximating stage problems are discretizations of the true stage problems that become more accurate as the discretizations become finer. This enables us to obtain approximate solutions of the stochastic decision problem.

A number of authors have looked to extend SDDP methods to deal with non-convex stage problems. The first approach replaces the non-convex components of the problem with convex approximations, for example using McCormick envelopes to approximate the production function of hydro plants as in [3]. The second approach convexifies the value function, e.g. using Lagrangian relaxation techniques. A recent paper [17] by Thomé et al. proposes a Lagrangian relaxation of the SDDP subproblem, and then uses the Lagrange multipliers to produce valid Benders cuts. A similar approach is adopted by Steeger and Rebennack in [16]. Abgottspon et al [1] introduce a heuristic to add locally valid cuts that enhance a convexified approximation of the value function. More recently Zou et al [19] prove the almost sure convergence of their SDDiP algorithm, which extends SDDP to problems with binary state variables. This method has recently been applied to a version of the river-chain optimization problem by Hjelmeland et al [11].

When the non-convexity in the Bellman function arises from state variables appearing in the stage objective function, as prices do in the reservoir optimization above, it is common to represent the price dynamics by a Markov chain in which each price state has its own separate cutting plane approximation for the Bellman function with price fixed (see e.g. [2] and [8]). MIDAS provides a different approximation to this discretization of the Bellman function. We should also note that the linear price dynamics in the reservoir optimization problem yields a value function that is concave in storage and convex in price, which enables one to linearly interpolate the Bellman function in the price dimension. This approach is investigated by Downward et al in a recent paper [6]. Since it exploits the specific strucure of the Bellman function, the performance of this method on single reservoir problems is better than MIDAS (as reported in the thesis of Wahid [18]). This is not surprising given that MIDAS is a general approach to solving these problems that does not exploit the saddle form of the Bellman function.

The rest of the paper is laid out as follows. We begin by outlining the approximation of $V_t(x)$ that is used by MIDAS. In section 3 we prove the convergence of MIDAS to a $2T\varepsilon$-optimal solution of a multistage deterministic optimization problem. Lastly, in section 4, we extend our proof to a sampling-based algorithm applied to the multistage stochastic optimization problem with T stages, and demonstrate its almost-sure convergence to a $2T\varepsilon$-optimal solution. A simple hydro-electric scheduling example illustrating the algorithm is presented in section 5. We conclude the paper in section 6 by summarizing the main contributions of the paper.

2 Static problem

Consider first the case of a static problem: $T = 1$, in a deterministic setting. The problem can then be written in the form

$$\max_{u \in U} r(u) + R(f(u)). \tag{4}$$

where r is an immediate reward and R can be interpreted as a terminal value function (as in the previous section). Here the decision set U is a compact (possibly discrete) subset of $\mathbb{R}^M$. With $u \in U$ is associated the "state" $f(u)$, $f : U \to X$, where X is the set of feasible states, a compact subset of $\mathbb{R}^N$. The cost functions are

$r : U \to \mathbb{R}$ and $R : X \to \mathbb{R}$, with r and $R \circ f$ upper semicontinuous, Denote by $\mathbf{1}$ a vector with all components equal to 1. We assume the existence of $\delta > 0$, $\varepsilon \geq 0$, such that

$$R(x) \leq R(y) + \varepsilon, \quad \text{if } x \leq y + \delta\mathbf{1}, \quad \text{for all } x, y \text{ in } X. \tag{5}$$

Remark 1 (i) If R is nondecreasing then (5) implies that R does not increase by more than ε for any change in x with ℓ_∞ norm smaller than δ. We need not assume that R is nondecreasing, as long as ε takes a sufficiently large value to satisfy (5). So we call the above condition an *approximate nondecreasing condition.*
(ii) If U (and hence X) is a discrete set, and R is nondecreasing, then (5) is satisfied by taking δ less than the distance between two distinct points of X, and $\varepsilon = 0$.
(iii) Since X is compact, if R is continuous and nondecreasing, since R is uniformly continuous, for any $\varepsilon > 0$, we can find $\delta > 0$ such that (5) holds.
(iv) The parameter ε will play an important role in the MIDAS algorithm. As shown in Figure 1 and Figure 2 it gives a measure of how far a step function that approximates R from above is permitted to deviate below R. This then determines the accuracy of an approximate upper bound for the optimal value of the problem we are solving.

Before stating an algorithm, let us explain how to compute an upper-bound approximation of the function $R(x)$. Let $\bar{R} \geq \max\{R(x); \ x \in X\}$. Given a sequence x^k in X, with $k = 1$ to H, set

$$q^k := R(x^k). \tag{6}$$

Define the set

$$\Omega^k = \{(x^h, q^h) : h = 1, 2, \ldots, k - 1\}. \tag{7}$$

The set of *supporting indices* (at step k) of $x \in X$ is defined as

$$\mathcal{H}^k(x) := \{h : 1 \leq h < k, \ x_i < x_i^h + \delta, \quad i = 1, \ldots, N\}. \tag{8}$$

Consider also the following function, that may be viewed as an approximation of the function R (see the lemma below):

$$Q^k(x) = \begin{cases} \bar{R} & \text{if } \mathcal{H}^k(x) \text{ is empty,} \\ \min\left\{q^h : h \in \mathcal{H}^k(x)\right\} & \text{otherwise.} \end{cases} \tag{9}$$

Note the strict inequality in (8); by virtue of this, the function $Q^k(x)$ is upper semicontinuous.

Lemma 1 *(i) The function Q^k is nondecreasing and upper semicontinuous in x, and nonincreasing in k:*

$$Q^{k+1}(x) \leq Q^k(x), \quad x \in X. \tag{10}$$

(ii) We have that

$$R(x) \leq Q^k(x) + \varepsilon, \quad \text{for any } x \in X, \tag{11}$$

$$k \in \mathcal{H}^{k+1}(y), \quad \text{for any } y \text{ with } y_i < x_i^k + \delta, \quad i = 1, \ldots, N, \tag{12}$$

$$Q^{k+1}(x^k) \leq q^k, \quad \text{for any } 1 \leq k \leq H. \tag{13}$$

Proof (i) That Q^k is nondecreasing as a function of x, follows from the fact that $\mathcal{H}^k(x)$ is itself nonincreasing as a function of x. That Q^k is nonincreasing as a function of k follows from the fact that $\mathcal{H}^k(x)$ is itself nondecreasing as a function of k. Upper semicontinuity of Q^k follows directly from its definition.
(ii) If $\mathcal{H}^k(x)$ is empty, (11) obviously holds, since then $Q^k(x) = \bar{R} \geq R(x)$. Assume now that $\mathcal{H}^k(x)$ is not empty. Let $x \in X$ and $h \in \mathcal{H}^k(x)$. By (5) and the definition of $\mathcal{H}^k(x)$, we get

$$R(x) - \varepsilon \leq R(x^h) = q^h. \tag{14}$$

Minimizing over $k \in \mathcal{H}^k(x)$ and using (9) yields (11). If $y_i < x_i^k + \delta$, $i = 1, \ldots, N$, then (12) follows directly from (8), and applying (9) then gives (13). $\qquad\square$

Two examples of the approximation of $R(x)$ by $Q^k(x)$ are given in Figure 1 and Figure 2.
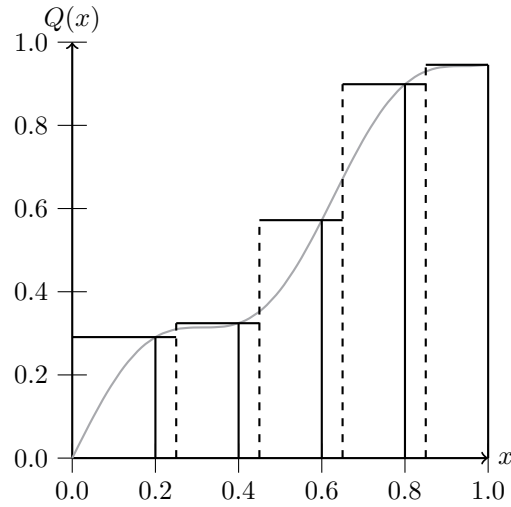

Fig. 1 Approximation of $R(x) = x + 0.1\sin(10x)$ shown in grey, by piecewise constant $Q^k(x)$, shown in black. Here $\delta = 0.05$ and $x^h = 0.2, 0.4, 0.6, 0.8$.
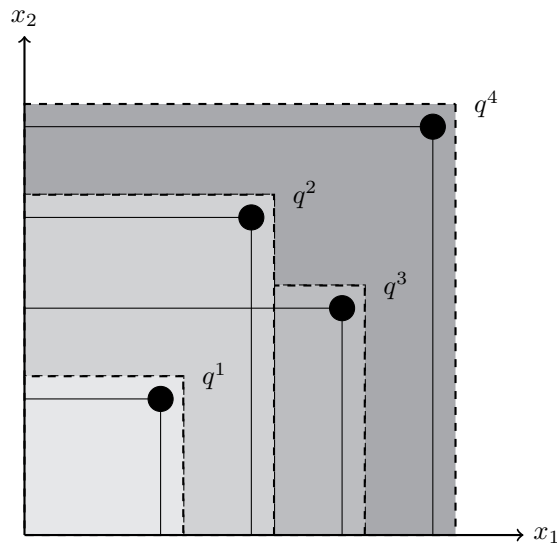

Fig. 2 Contour plot of $Q^k(x)$ when $k = 4$. The circled points are x^h, $h = 1, 2, 3, 4$. The darker shading indicates increasing values of $Q^k(x)$ (which equals $R(x^h)$ in the region containing x^h.)

The definition of Q^k in terms of supporting indices is made for notational convenience. In practice $Q^k(x)$ is computed using mixed integer programming (hence the name MIDAS). We propose one possible formulation for doing this in the Appendix.

We propose the following algorithm:

Algorithm 1 Static MIDAS

1. Set $k = 1$, $\Omega^k = \emptyset$, and $Q^k(x) = \bar{R}$.
2. Forward pass:
   (a) Solve $\max_{u \in U} \left\{ r(u) + Q^k(f(u)) \right\}$ to give u^k;
   (b) If $\left\| f(u^k) - x^h \right\|_\infty < \delta$ for some $1 < h < k$, then set $x^k := x^h$, else set $x^k := f(u^k)$.
3. Backward pass: $\Omega^{k+1} := \Omega^k \cup \left\{ (x^k, q^k) \right\}$, where $q^k := R(x^k)$.
4. Stopping test: Set $\varepsilon_k := Q^k(f(u^k)) - Q^{k+1}(f(u^k))$. If $\varepsilon_k \leq \bar{\varepsilon}$, set $H := k$, and stop.
5. Increase k by 1 and go to step 2.

Remark 2 (i) The maximum in step 2(a) is attained if U is a discrete set, and more generally if f is continuous, since r and Q^k are upper semicontinuous, the latter as a consequence of lemma 1.

(ii) In general, f is an arbitrary continuous function, so step 2(a) requires the solution of a global optimization problem. In practice Q^k is defined by a MIP (as defined in the Appendix). If r and f are affine functions of x and u then the optimization in step 2(a) also becomes a MIP. This makes the practical application of the algorithm straightforward.

(ii) Note that $\varepsilon_k \geq 0$.

Theorem 1 *(i) At any iteration k, u^k is a $(2\varepsilon + \varepsilon_k)$-solution of problem (4).*
(ii) The stopping test is activated after finitely many iterations.

Proof (i) Denote by V the value of problem (4). Since $\|x^k - f(u^k)\|_\infty < \delta$, (12) implies that

$$k \in \mathcal{H}^{k+1}(f(u^k)). \tag{15}$$

Therefore

$$
\begin{aligned}
V &= \max_{u \in U} \left(r(u) + R(f(u)) \right) && \text{definition of } V \\
&\leq \max_{u \in U} \left(r(u) + Q^k(f(u)) \right) + \varepsilon && \text{by (11)} \\
&= r(u^k) + Q^k(f(u^k)) + \varepsilon && \text{definition of } u^k \\
&= r(u^k) + Q^{k+1}(f(u^k)) + \varepsilon + \varepsilon_k && \text{definition of } \varepsilon_k \text{ in the algorithm} \\
&\leq r(u^k) + q^k + \varepsilon + \varepsilon_k && \text{consequence of (15) and (9)} \\
&\leq r(u^k) + R(f(u^k)) + 2\varepsilon + \varepsilon_k && \text{consequence of (5) and } \|x^k - f(u^k)\|_\infty < \delta.
\end{aligned}
$$

(ii) Since the algorithm generates finitely many points, after finitely many iterations, $Q^{k+1} = Q^k$, and consequently $\varepsilon_k = 0$. The conclusion follows.   □

Remark 3 (i) Once convergence of the algorithm is obtained, so that a $2\varepsilon + \varepsilon_H$-solution has been found, one can choose to continue the computations with a smaller value of ε. This may be efficient for avoiding small steps that would occur when taking a small value of ε at first.

(ii) If the set X is discrete and if R is nondecreasing, taking $\delta > 0$ less than the ℓ^∞ distance between two distinct points in X, we can take $\varepsilon = 0$. The output of the above algorithm is then a solution of (4).

3 Multistage optimization problems

In this section we describe the MIDAS algorithm for a deterministic multistage optimization problem, and prove its convergence. Given an initial state $\bar{x}$, we consider a deterministic optimization problem of the form:

$$
\begin{aligned}
\text{MP:} \quad \max_{x,u} \quad & \sum_{t=\tau}^{T} r_t(x_t, u_t) + R(x_{T+1}) \\
\text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t), \quad t = \tau, \ldots, T, \\
& x_1 = \bar{x}, \\
& u_t \in U_t(x_t), \ x_t \in X_t, \quad t = \tau, \ldots, T.
\end{aligned}
$$

Here $1 \leq \tau \leq T + 1$. We have the state constraint $x_t \in X_t$, where X_t is a compact subset of $\mathbb{R}^N$, and for each $x_t \in X_t$, $U_t(x)$ is a compact subset of $\mathbb{R}^M$. The multimapping from X_t to the set of subsets of $\mathbb{R}^m$, $x \mapsto U_t(x)$ is assumed to be compatible with the state constraint, in the sense that $f_t(x_t, u_t) \in X_{t+1}$, whenever $x_t \in X_t$ and $u_t \in U_t(x_t)$. We denote by $V_\tau(x_\tau)$ the value of the above problem with initial value of the state equal to x_τ. It is well known that V_t satisfies the dynamic programming principle

$$
\begin{cases}
V_t(x) &= \sup_{u \in U_t(x)} \{r_t(x, u) + V_{t+1}(f_t(x, u))\}, \quad t = 1, \ldots, T, \\
V_{T+1}(x) &= R(x).
\end{cases}
\tag{16}
$$

Let $\bar{V}$ be an upper bound on $V_t(x)$, $t = 1, 2, \ldots, T + 1$. We assume that there exist $\delta > 0$ and $\varepsilon \geq 0$ such that the following approximate nondecreasing condition holds:

$$
V_t(x) \leq V_t(y) + \varepsilon, \quad \text{if } x \leq y + \delta \mathbf{1}, \quad \text{for all } x, y \text{ in } X_t, t = 2 \text{ to } T + 1.
\tag{17}
$$

Note that for $t = 1$ we need no such condition, since we need to estimate V_1 only at the point $\bar{x}$. In the multistage case, we approximate each stage value function $V_t(x)$ with a piecewise constant function $Q_t^k(x)$. At each stage t we have a sequence $x_t^k \in X_t$, $k = 1, \ldots, H$. We will define the associated numbers q_t^k later, and set

$$
\Omega_t^k = \{(x_t^h, q_t^h) : h = 1, \ldots, k - 1\}, \quad \text{for all } t = 1, \ldots, T + 1.
$$

Given these we now define the set of supporting indices

$$
\mathcal{H}_t^k(x) := \{1 \leq h < k; \ x_{ti} < x_{ti}^h + \delta, \ i = 1, \ldots, N\},
\tag{18}
$$

as well as the following approximations of the Bellman functions:

$$
Q_t^k(x) = \begin{cases}
\bar{V} & \text{if } \mathcal{H}_t^k(x) \text{ is empty,} \\
\min \left\{ q_t^h : h \in \mathcal{H}_t^k(x) \right\} & \text{otherwise.}
\end{cases}
\tag{19}
$$

We will make use of the following simple result.

Lemma 2 *If $h < k$ and $\left\| x - x_t^h \right\|_\infty < \delta$ then $Q_t^k(x) \leq q_t^h$.*

Proof We have $x_i < x_{ti}^h + \delta$, $i = 1, \ldots, N$, so $h \in \mathcal{H}_t^k(x)$. Thus $Q_t^k(x) \leq q_t^h$ as required. □

The deterministic MIDAS algorithm (Algorithm 2) generates a sequence of functions $Q_t^k(x)$, $t = 1, \ldots, T + 1$. For each k we define:

the *forward-step* decision:

$$
u_t^k \in \arg \max_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^k(f_t(x_t^k, u)) \right\};
\tag{20}
$$

the *backward-step* decision:

$$\hat{u}_t^k \in \arg \max_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^{k+1}(f_t(x_t^k, u)) \right\}; \qquad (21)$$

the *optimal decision*:

$$\tilde{u}_t^k \in \arg \max_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + V_{t+1}(f_t(x_t^k, u)) \right\}; \qquad (22)$$

the *stage error*:

$$\varepsilon_{k,t} := Q_{t+1}^k(f_t(x_t^k, u_t^k)) - Q_{t+1}^{k+1}(f_t(x_t^k, u_t^k)), \quad t = 1, \dots, T. \qquad (23)$$

Note that $\varepsilon_{k,t} \geq 0$. Given $\bar{\varepsilon} \geq 0$, the stopping test is

$$\sum_{t=1}^{T} \varepsilon_{k,t} \leq \bar{\varepsilon}. \qquad (24)$$

We can now write down the steps of the algorithm as follows.

Algorithm 2 Deterministic MIDAS

---

1. Choose $\bar{\varepsilon} \geq 0$, and set $k = 1$, $\Omega_t^k = \emptyset$, and $Q_t^k(x) = \bar{V}$ an upper bound on $V_t(x)$, $t = 1, 2, \dots, T + 1$.
2. Forward pass: Set $x_1^k = \bar{x}$. For $t = 1$ to T,
    (a) Solve $\max\limits_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^k(f_t(x_t^k, u)) \right\}$ to give u_t^k;
    (b) If $\left\| f_t(x_t^k, u_t^k) - x_{t+1}^h \right\|_\infty < \delta$ for $h < k$ then set $x_{t+1}^k = x_{t+1}^h$, else set $x_{t+1}^k = f_t(x_t^k, u_t^k)$.
3. Backward pass: Set $\Omega_{T+1}^{k+1} = \Omega_{T+1}^k \cup \left\{ (x_{T+1}^k, q_{T+1}^k) \right\}$ where $q_{T+1}^k := R(x_{T+1}^k)$.
    For every $t = T$ down to 1,
    (a) Compute $q_t^k = \max\limits_{u \in U_t(x_t^k)} \left\{ r_t(x_t^k, u) + Q_{t+1}^{k+1}(f_t(x_t^k, u)) \right\}$;
    (b) Set $\Omega_t^{k+1} = \Omega_t^k \cup \left\{ (x_t^k, q_t^k) \right\}$.
4. Stopping test: if (24) holds, set $H := k$, and stop.
5. Increase k by 1 and go to step 2.

Observe that step 2 of Algorithm 2 does not typically generate a feasible state trajectory since step 2(b) can jump to a nearby state x_{t+1}^h in Ω_{t+1}^k. We call the resulting sequence of states $\{x_{t+1}^k, t = 1 \dots, T\}$ a *pseudo trajectory*.

Lemma 3 *For all iterations k,*

$$\begin{cases} \text{(i)} \ \ Q_t^k(x) \geq V_t(x) - (T + 2 - t)\varepsilon, \ \text{for all } x \in X_t, \ t = 2, \dots, T + 1, \\ \text{(ii)} \ \ q_1^k \geq V_1(\bar{x}) - T\varepsilon. \end{cases} \qquad (25)$$

Proof (i) We prove (25)(i) by backward induction. For $t = T + 1$, the result follows from Lemma 1 since $V_{T+1}(x) = R(x)$ for all $x \in X_{T+1}$. Now let (25) hold for some $2 < t \leq T + 1$. It follows that for every k

$$\begin{aligned} q_{t-1}^k &= \max_{u \in U_{t-1}(x_{t-1}^k)} \left\{ r_{t-1}(x_{t-1}^k, u) + Q_t^{k+1}(f_{t-1}(x_{t-1}^k, u)) \right\} && \text{definition of } q_{t-1}^k \\ &\geq r_{t-1}(x_{t-1}^k, \tilde{u}_{t-1}^k) + Q_t^{k+1}(f_{t-1}(x_{t-1}^k, \tilde{u}_{t-1}^k)) && \text{definition of a maximum} \\ &\geq r_{t-1}(x_{t-1}^k, \tilde{u}_{t-1}^k) + V_t(f_{t-1}(x_{t-1}^k, \tilde{u}_{t-1}^k)) - (T + 2 - t)\varepsilon && \text{induction hypothesis} \\ &= V_{t-1}(x_{t-1}^k) - (T + 2 - t)\varepsilon. && \text{definition of } \tilde{u}_{t-1}^k \end{aligned} \qquad (26)$$

Now let $x \in X_{t-1}$. If $\mathcal{H}_{t-1}^k(x)$ is empty then $Q_{t-1}^k(x) = \bar{V}$, so (25) holds trivially. Otherwise suppose $h \in \mathcal{H}_{t-1}^k(x)$, so $x_i < x_{t-1,i}^h + \delta$, $i = 1$ to N. If $t \geq 3$, by (17) and (26), we get that

$$V_{t-1}(x) \leq V_{t-1}(x_{t-1}^k) + \varepsilon \leq q_{t-1}^h + (T + 3 - t)\varepsilon. \tag{27}$$

Minimizing over $h \in \mathcal{H}_{t-1}^k(x)$ and using (9) we obtain that (25) holds for $t - 1$. Point (i) follows.
(ii) In step (a) of this proof, we have obtained (26) also when $t = 2$, which upon setting $x_1^k = \bar{x}$ becomes

$$q_1^k \geq V_1(\bar{x}) - T\varepsilon, \tag{28}$$

so (25)(ii) follows. $\qquad\square$

Lemma 4 *(i) For each iteration k,*

$$V_1(\bar{x}) \leq \sum_{t=1}^{T} r_t(x_t^k, u_t^k) + R(x_{t+1}^k) + T\varepsilon + \sum_{t=1}^{T} \varepsilon_{k,t}. \tag{29}$$

(ii) The algorithm terminates after finitely many iterations at iteration $k = H$, and the resulting pseudo trajectory satisfies

$$V_1(\bar{x}) \leq \sum_{t=1}^{T} r_t(x_t^H, u_t^H) + R(x_{t+1}^H) + T\varepsilon + \bar{\varepsilon}. \tag{30}$$

Proof (i) After step 2(b) of iteration k we have, for $t = 1$ to T:

$$\left\| x_{t+1}^k - f_t(x_t^k, u_t^k) \right\|_{\infty} < \delta.$$

So, $f_t(x_t^k, u_t^k) < x_{t+1}^k + \delta \mathbf{1}$. This implies

$$k \in \mathcal{H}_t^{k+1}(f_t(x_t^k, u_t^k)). \tag{31}$$

We also have that, for $t = 1$ to T:

$$
\begin{aligned}
q_t^k &= r_t(x_t^k, \hat{u}_t^k) + Q_{t+1}^{k+1}(f_t(x_t^k, \hat{u}_t^k)) && \text{by (21) and the definition of } q_t^k \\
&\leq r_t(x_t^k, \hat{u}_t^k) + Q_{t+1}^k(f_t(x_t^k, \hat{u}_t^k)) && \text{by the monotonicity of } k \mapsto Q_{t+1}^k \\
&\leq r_t(x_t^k, u_t^k) + Q_{t+1}^k(f_t(x_t^k, u_t^k)) && \text{by the definition of } u_t^k \text{ in (20)} \\
&= r_t(x_t^k, u_t^k) + Q_{t+1}^{k+1}(f_t(x_t^k, u_t^k)) + \varepsilon_{k,t} && \text{by the definition of } \varepsilon_{k,t} \\
&\leq r_t(x_t^k, u_t^k) + q_{t+1}^k + \varepsilon_{k,t} && \text{by (31)}.
\end{aligned}
$$

Also,

$$q_{T+1}^k = R(x_{T+1}^k). \tag{32}$$

Summing the previous inequalities and equality, we get that

$$q_1^k \leq \sum_{t=1}^{T} r_t(x_t^k, u_t^k) + R(x_{T+1}^k) + \sum_{t=1}^{T} \varepsilon_{k,t}. \tag{33}$$

By Lemma 3 for $t = 1$, we have that

$$V_1(\bar{x}) \leq q_1^k + T\varepsilon. \tag{34}$$

The result follows by combining (33) and (34).
(ii) The algorithm visits finitely many points. So, by backward induction over t, we get that the pairs (x_t^k, q_t^k) also take finitely many values. So, for large enough k, if the algorithm does not stop, the functions Q_t^k do not depend on k, and therefore all $\varepsilon_{k,t} = 0$. But this contradicts the fact that the stopping test is never satisfied. $\qquad\square$

The previous result applies to the sequence (x_t^H, u_t^H) which satisfies

$$\|x_{t+1} - f_t(x_t, u_t)\|_\infty < \delta, \quad t = 1, \dots, T. \tag{35}$$

Assume that with any such pseudo trajectory, we can associate a true trajectory by increasing the cost by some nonnegative amount c_δ. Denote by $(\bar{x}_t^H, \bar{u}_t^H)$, for $t = 1$ to $T + 1$, such a trajectory associated with (x^H, u^H). We get then the following corollary:

Corollary 1 *Set $\hat{\varepsilon}_H := T\varepsilon + \sum_{t=1}^T \varepsilon_{H,t} + c_\delta$. We have that, at each iteration $1 \le k \le M$: of the algorithm:*

$$V_1(\bar{x}) \le \sum_{t=1}^T r_t(\bar{x}_t^H, \bar{u}_t^H) + R(\bar{x}_{t+1}^H) + \hat{\varepsilon}_H. \tag{36}$$

In particular, the output trajectory $(\bar{x}^H, \bar{u}^H)$ is $T\varepsilon + \bar{\varepsilon} + c_\delta$ optimal.

The previous result is rather weak in that it gives no bounds of the size of c_δ. If we set $\bar{\varepsilon} = 0$, then we can obtain a bound on optimality for the trajectory obtained when the algorithm terminates. This will occur at iteration $k = H$ when for every $t = 1, 2, \dots, T$,

$$Q_{t+1}^H(f_t(x_t^H, u_t^H)) - Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) = 0, \quad t = 1, \dots, T. \tag{37}$$

This means that the forward pass in iteration $H + 1$ will visit the same points as in iteration H, so

$$(x_t^{H+1}, u_t^{H+1}) = (x_t^H, u_t^H), \, t = 1, 2, \dots, T + 1,$$

and so the functions Q_{t+1}^H and Q_{t+1}^{H+1} will be identical for all $t = 1, 2, \dots, T + 1$. It follows that in the forward pass, u_t^H will maximize $r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H))$ giving

$$r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \ge r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)),$$

and (21) implies

$$r_t(x_t^H, u_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \le r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)),$$

so

$$r_t(x_t^H, u_t^H) + Q_{t+1}^H(f_t(x_t^H, u_t^H)) = r_t(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)). \tag{38}$$

We now can establish the following result.

Lemma 5 *For every $t = 1, 2, \dots, T$,*

$$Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \le V_{t+1}(f_t(x_t^H, u_t^H)) + (T - t + 1)\varepsilon.$$

Proof We proceed by backwards induction on t. When $t = T$, after step 2(b) of iteration H we have for some $h \le H$

$$\left\| x_{T+1}^h - f_T(x_T^H, u_T^H) \right\|_\infty < \delta$$

so by Lemma 2

$$\begin{aligned} Q_{T+1}^{H+1}(f_T(x_T^H, u_T^H)) &\le q_{T+1}^h \\ &= V_{T+1}(x_{T+1}^h) \\ &\le V_{T+1}(f(x_T^H, u_T^H)) + \varepsilon, \end{aligned}$$

by (17).

Assume as an inductive hypothesis that

$$Q_{t+1}^{H+1}(f_t(x_t^H, u_t^H)) \leq V_{t+1}(f(x_t^H, u_t^H)) + (T - t + 1)\varepsilon$$

We show that

$$Q_t^{H+1}(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) \leq V_t(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) + (T - t + 2)\varepsilon.$$

Since at termination of the algorithm

$$\|x_t^H - f_{t-1}(x_{t-1}^H, u_{t-1}^H)\|_\infty < \delta,$$

it follows that

$$
\begin{aligned}
Q_t^{H+1}(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) &\leq q_t^H && \text{by Lemma 2} \\
&= r(x_t^H, \hat{u}_t^H) + Q_{t+1}^{H+1}(f_t(x_t^H, \hat{u}_t^H)) && \text{by (21)} \\
&= r(x_t^H, u_t^H) + Q_{t+1}^H(f_t(x_t^H, u_t^H)) && \text{by (38)} \\
&\leq r(x_t^H, u_t^H) + V_{t+1}(f_t(x_t^H, u_t^H)) + (T - t + 1)\varepsilon && \text{induction hypothesis} \\
&\leq r(x_t^H, \tilde{u}_t^k) + V_{t+1}(f_t(x_t^H, \tilde{u}_t^k)) + (T - t + 1)\varepsilon && \text{optimality of } \tilde{u}_t^k \\
&= V_t(x_t^H) + (T - t + 1)\varepsilon && \text{definition of } V_t \\
&\leq V_t(f_{t-1}(x_{t-1}^H, u_{t-1}^H)) + (T - t + 2)\varepsilon
\end{aligned}
$$

where the last inequality follows from $\|x_t^H - f_{t-1}(x_{t-1}^H, u_{t-1}^H)\|_\infty < \delta$ and (17). This establishes the result for $t - 1$ and hence all t by induction.   □

Using Lemma 5, we can show that the first-stage decision u_1^H obtained when Algorithm 2 terminates (at iteration H where $\varepsilon^H = 0$) is $2T\varepsilon$-optimal.

Theorem 2 *Suppose $\bar{\varepsilon} = 0$. Upon termination of the algorithm the first-stage decision u_1^H satisfies*

$$r_1(\bar{x}, u_1^H) + V_2(f_1(\bar{x}, u_1^H)) \geq V_1(\bar{x}) - 2T\varepsilon.$$

Proof For the optimal first stage decision u_1^*, (25)(ii) gives

$$r_1(\bar{x}, u_1^*) + Q_2^H(f_1(\bar{x}, u_1^*)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - T\varepsilon$$

and by (20)

$$r_1(\bar{x}, u_1^H) + Q_2^H(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + Q_2^H(f_1(\bar{x}, u_1^*))$$

so

$$r_1(\bar{x}, u_1^H) + Q_2^H(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - T\varepsilon. \tag{39}$$

Now upon termination of the algorithm (38) gives

$$Q_2^H(f_1(\bar{x}, u_1^H)) = Q_2^{H+1}(f_1(\bar{x}, u_1^H)), \tag{40}$$

and Lemma 5 implies

$$Q_2^{H+1}(f_1(\bar{x}, u_1^H)) \leq V_2(f_1(\bar{x}, u_1^H)) + T\varepsilon, \tag{41}$$

so (39), (40), and (41) yield

$$r_1(\bar{x}, u_1^H) + V_2(f_1(\bar{x}, u_1^H)) \geq r_1(\bar{x}, u_1^*) + V_2(f_1(\bar{x}, u_1^*)) - 2T\varepsilon,$$

giving the result.   □

4 Multistage stochastic optimization problems

4.1 Setting

We extend model MP, described in Section 3, to include random noise on the state transition function. We model the realizations of the noise in the form of a *scenario tree* with nodes $n \in \mathcal{N}$ and leaves in $\mathcal{L}$, where n represents different future states of the world. Each node n has a probability $p(n)$. By convention we number the root node $n = 0$ (with $p(0) = 1$). The unique predecessor of node $n \neq 0$ is denoted by $n-$. We denote the set of children of node $n \in \mathcal{N} \setminus \mathcal{L}$ by $n+$, and let $M_n = |(n+)|$. The depth $d(n)$ of node n is the number of nodes on the path from node n to node 0, so $d(0) = 1$ and we assume that every leaf node has the same depth, say $d_{\mathcal{L}}$. The depth of a node can be interpreted as a time index, so we can identify $d_{\mathcal{L}}$ with time $T + 1$ as defined in section 3. Set $\mathcal{N}_* := \mathcal{N} \setminus \{0\}$. The formulation of MSP in the scenario tree is

$$\text{MSPT: max} \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n(x_{n-}, u_n) + \sum_{n \in \mathcal{L}} p(n) R(x_n)$$
$$\text{s.t.} \quad x_n = f_n(x_{n-}, u_n), \qquad\qquad n \in \mathcal{N}_*,$$
$$x_0 = \overline{x},$$
$$u_n \in U_n(x_{n-}), \qquad\qquad n \in \mathcal{N}_*,$$
$$x_n \in X_n, \qquad\qquad n \in \mathcal{N}_*.$$

The above probabilities must satisfy

$$p(n) \geq 0, \ n \in \mathcal{N}; \quad \sum_{n \in \mathcal{L}} p(n) = 1; \quad p(n) = \sum_{m \in n+} p(m), \quad n \in \mathcal{N} \setminus \mathcal{L}. \tag{42}$$

Observe in MSPT that we have a choice between hazard-decision and decision-hazard formulations that was not relevant in the deterministic problem. To be consistent with most implementations of SDDP, we have chosen a hazard-decision setting. This means that u is chosen in node n after the information from node n is revealed. The dynamic programming principle for MSPT can be expressed as

$$\begin{cases} V_n(x_n) = \sum_{m \in n+} \frac{p(m)}{p(n)} \max_{u \in U_m(x_n)} \{r_m(x_n, u) + V_m(f_m(x_n, u))\}, & n \in \mathcal{N} \setminus \mathcal{L}, \\ V_n(x_n) = R(x_n), & n \in \mathcal{L}. \end{cases}$$

We seek a policy that maximizes $V_0(\overline{x})$. Below we give two different extensions of the MIDAS algorithm of the deterministic setting; *Full-tree MIDAS* goes over every node at each iteration; and *Sampled MIDAS* computes only one pseudo trajectory at each iteration.

Now given a scenario tree, we approximate each node value function $V_n(x)$, $n \in \mathcal{N}_*$, with a piecewise constant function $Q_n^k(x)$, where $k = 1, 2, \ldots, H$ is the algorithm iteration. We have a sequence x_n^k in X_n. Associated with each x_n^k is a value q_n^k. For leaf nodes we have that

$$q_n^k := R(x_n^k), \quad n \in \mathcal{L}. \tag{43}$$

As before we define

$$\Omega_n^k = \{(x_n^k, q_n^k) : h = 1, \ldots, k - 1\}, \quad \text{for all } n \in \mathcal{N}_* \setminus \mathcal{L}$$

We now define the *supporting indices* of $x \in X_n$ as

$$\mathcal{H}_n^k(x) := \{1 \leq h < k; \ x_i < x_{n,i}^k + \delta, \ i = 1, \ldots, N\}, \tag{44}$$

as well as the following function:

$$Q_n^k(x) = \begin{cases} \bar{V} & \text{if } \mathcal{H}_n^k(x) \text{ is empty,} \\ \min\{q_n^k : h \in \mathcal{H}_n^k(x)\} & \text{otherwise.} \end{cases} \tag{45}$$

and as before we suppress the dependence of $\mathcal{H}_n^k(x)$ and $Q_n^k(x)$ on the parameter δ. The following result is immediate.

Lemma 6 *If $h < k$ and $\left\| x - x_n^h \right\|_\infty < \delta$ then $Q_n^k(x) \leq q_n^h$.*

4.2 Full-tree MIDAS algorithm

The stopping test is based on the following amounts:

$$
\begin{cases}
\varepsilon_{k,m} := Q_m^k(f_m(x_n^k, u_m^k)) - Q_m^{k+1}(f_m(x_n^k, u_m^k)), & \text{for all } n \in \mathcal{N} \setminus \mathcal{L} \text{ and } m \in n+, \\
\varepsilon_k := \displaystyle\sum_{m \in \mathcal{N}_*} p(m)\varepsilon_{k,m}.
\end{cases}
\tag{46}
$$

Algorithm 3 Full tree MIDAS

Set $k = 1$, and $\Omega_n^k = \emptyset$, for all $n \in \mathcal{N}$.

1. Forward pass: Set $x_0^k = \overline{x}$, and $n = 0$. While $n \notin \mathcal{L}$, for each $m \in n+$:
   (a) Solve $\displaystyle\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u)) \right\}$ to give u_m^k;
   (b) If $\left\| f_m(x_n^k, u_m^k) - x_m^k \right\|_\infty < \delta$ for some $h < k$ then set $x_m^k = x_m^k$. Otherwise, set $x_m^k = f_m(x_n^k, u_m^k)$.
   (c) Set $n = m$.
2. Backward pass:
   (a) For every $n \in \mathcal{L}$, set
   $$
   \Omega_n^{k+1} := \Omega_n^k \cup \{(x_n^k, q_n^k)\}, \quad \text{where } q_n^k = R(x_n^k).
   \tag{47}
   $$
   (b) 'In order of decreasing depth', for each node n
      i. Compute
      $$
      q_n^k = \sum_{m \in n+} \frac{p(m)}{p(n)} \left[\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\} \right].
      \tag{48}
      $$
      with maxima in computation of q_n^k attained at $\hat{u}_m^k$, for each $m \in n+$.
      ii. Set $\Omega_n^{k+1} := \Omega_n^k \cup \{(x_n^k, q_n^k)\}$.
3. If ε_k, defined in (46), satisfies $\varepsilon_k \leq \overline{\varepsilon}$, set $H := k$ and stop.
4. Increase k by 1 and go to step 1.

We assume that there exist $\delta > 0$ and $\varepsilon > 0$ such that, for any $n \in \mathcal{N}_*$, the following approximate nondecreasing condition holds:
$$
V_n(x) \leq V_n(y) + \varepsilon, \quad \text{if } x \leq y + \delta\mathbf{1}, \quad \text{for all } x, y \text{ in } X_n.
\tag{49}
$$

Lemma 7 *For every $n \in \mathcal{N}_*$, and for all iterations k*
$$
Q_n^k(x) \geq V_n(x) - (d_\mathcal{L} + 1 - d(n))\varepsilon.
\tag{50}
$$

Proof For a leaf node m in $\mathcal{L}$, with depth $d(m) = d_\mathcal{L}$, (50) follows from Lemma 1. Now let $n \in \mathcal{N}_* \setminus \mathcal{L}$ have depth $d(n)$, and suppose as an inductive hypothesis that (50) holds for all nodes with depth greater than $d(n)$. Let $x \in X_n$, and $(h \in \mathcal{H}_n^k(x)$, for some $h < k$. Let

$$
\tilde{u}_m^h \in \arg\max_{u \in U_m(x_n^h)} \left\{ r_m(x_n^h, u) + V_m\left(f_m(x_m^h, u)\right) \right\}, \quad \text{for any } m \in n+.
$$

Then

$$q_n^h = \sum_{m \in n+} \frac{p(m)}{p(n)} \max_{u \in U_m(x_n^h)} \left\{ r_m(x_n^h, u) + Q_m^{h+1}\left(f_m(x_n^h, u)\right)\right\}$$

$$\geq \sum_{m \in n+} \frac{p(m)}{p(n)} \left(r_m(x_n^h, \tilde{u}_m^h) + Q_m^{h+1}\left(f_m(x_n^h, \tilde{u}_m^h)\right)\right).$$

Applying (50) now yields

$$q_n^h \geq \sum_{m \in n+} \frac{p(m)}{p(n)}(r_m(x_n^h, \tilde{u}_m^h) + V_m\left(f_m(x_n^h, \tilde{u}_m^h)\right) - (d_{\mathcal{L}} + 1 - d(m))\varepsilon)$$

$$= V_n(x_n^h) - (d_{\mathcal{L}} + 1 - d(m))\varepsilon,$$

$$= V_n(x_n^h) - (d_{\mathcal{L}} - d(n))\varepsilon,$$

since $d(n) = d(m) - 1$. Now for $h \in \mathcal{H}_n^k(x)$, $x_i < x_{ni}^k + \delta$, $i = 1, 2, \ldots, N$. By (49) and the above inequality, we get that

$$V_n(x) \leq V_n(x_n^h) + \varepsilon \leq q_n^h + (d_{\mathcal{L}} + 1 - d(n))\varepsilon. \tag{51}$$

The conclusion follows by minimizing over $h \in \mathcal{H}_n^k(x)$. $\square$

We next analyze the convergence of the Full-tree MIDAS algorithm. As in the deterministic case, we define

$$u_m^k \in \arg \max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u))\right\}, \quad m \in n+, \tag{52}$$

$$\hat{u}_m^k \in \arg \max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u))\right\}, \quad m \in n+. \tag{53}$$

$$\tilde{u}_m^k \in \arg \max_{U_m(x_n^k)} \left\{ r_m(x_n^k, u) + V_m(f_m(x_n^k, u))\right\}, \quad m \in n+, \tag{54}$$

$$u_m^* \in \arg \max_{u \in U_m(\bar{x})} \left\{ r_m(\bar{x}, u) + V_m(f_m(\bar{x}, u))\right\}, \quad m \in n+, \tag{55}$$

Theorem 3 *i) We have, at each iteration k of the algorithm:*

$$V_0(\bar{x}) \leq \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n(x_{n-}^k, u_n^k) + \sum_{n \in \mathcal{L}} p(n) R\left(x_n^k\right) + (d_{\mathcal{L}} - 1)\varepsilon + \varepsilon_k. \tag{56}$$

ii) The stopping test is satisfied after a finite value number of steps (when $k = H$), and then, the current policy is such that

$$V_0(\bar{x}) \leq \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n(x_{n-}^H, u_n^H) + \sum_{n \in \mathcal{L}} p(n) R\left(x_n^H\right) + (d_{\mathcal{L}} - 1)\varepsilon + \bar{\varepsilon}. \tag{57}$$

Proof (i) Adapting the arguments for the deterministic case, we can write

$$p(n)q_n^k = \sum_{m \in n+} p(m) \left(r_m(x_n^k, \hat{u}_m^k) + Q_m^{k+1}(f_m(x_n^k, \hat{u}_m^k))\right)$$

$$\leq \sum_{m \in n+} p(m) \left(r_m(x_n^k, \hat{u}_m^k) + Q_m^k(f_m(x_n^k, \hat{u}_m^k))\right)$$

$$\leq \sum_{m \in n+} p(m) \left(r_m(x_n^k, u_m^k) + Q_m^k(f_m(x_n^k, u_m^k))\right)$$

$$= \sum_{m \in n+} p(m) \left(r_m(x_n^k, u_m^k) + Q_m^{k+1}(f_m(x_n^k, u_m^k)) + \varepsilon_{k,m}\right) \tag{58}$$

$$\leq \sum_{m \in n+} p(m) \left(r_m(x_n^k, u_m^k) + q_m^k + \varepsilon_{k,m}\right).$$

As a special case

$$q_0^k \leq \sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^k) + q_m^k + \varepsilon_{k,m} \right),$$

so substituting for $p(m)q_m^k$ recursively throughout the tree and using (58) and

$$p(n)q_n^k = p(n)R(x_n^k), \quad \text{for all } n \in \mathcal{L}, \tag{59}$$

yields

$$q_0^k \leq \sum_{n \in \mathcal{N}_*} p(n)r_n(x_{n-}, u_n^k) + \sum_{n \in \mathcal{L}} p(n)R\left(x_n^k\right) + \sum_{n \in \mathcal{N}_*} p(n)\varepsilon_{k,n} \tag{60}$$

On the other hand, Lemma 7 gives

$$
\begin{aligned}
V_0(\bar{x}) &= \sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) \right) \\
&\leq \sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^*) + Q_m^{k+1}(f_m(\bar{x}, u_m^*)) \right) + (d_{\mathcal{L}} - 1)\varepsilon \\
&\leq \sum_{m \in 0+} p(m) \left(r_m(\bar{x}, \hat{u}_m^k) + Q_m^{k+1}(f_m(\bar{x}, \hat{u}_m^k)) \right) + (d_{\mathcal{L}} - 1)\varepsilon \\
&= q_0^k + (d_{\mathcal{L}} - 1)\varepsilon.
\end{aligned}
\tag{61}
$$

The result follows from combining (60) and (61).

(ii) The result is a consequence of (i) and the fact that, as in the deterministic case, after finitely many iterations, we will have $\varepsilon_{k,n} = 0$ for each node n. □

As before, if we set $\bar{\varepsilon} = 0$, then at some iteration $k = H$ the forward pass in every iteration $k > H$ will visit the same points x_n^H as in iteration H, so for every $t = 1, 2, \ldots, T$,

$$Q_n^H(f_n(x_{n-}^H, u_n^H)) = Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) \tag{62}$$

This means that the functions Q_n^H and Q_n^{H+1} will be identical for all $n \in \mathcal{N}_*$. It follows that

$$r_n(x_{n-}, u_n^H) + Q_n^H(f_n(x_{n-}, u_n^H)) = r_n(x_{n-}^H, \hat{u}_n^H) + Q_n^{H+1}(f_n(x_{n-}, \hat{u}_n^H)) \tag{63}$$

We now can establish the following result.

Lemma 8 *For every $n \in \mathcal{N}_*$,*

$$Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) \leq V_n(f_n(x_{n-}^H, u_n^H)) + (d_{\mathcal{L}} - d(n) + 1)\varepsilon.$$

Proof We proceed by induction on n. When $n \in \mathcal{L}$, after step 2(b) of iteration H we have for some $h < H$

$$\left\| x_n^h - f_n(x_{n-}^H, u_n^H) \right\|_\infty < \delta$$

so by Lemma 6

$$
\begin{aligned}
Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) &\leq q_n^h \\
&= V_n(x_n^h) \\
&\leq Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) + \varepsilon.
\end{aligned}
$$

by (49).

Assume as an inductive hypothesis that for every m with $d_m = d$ we have

$$Q_m^{H+1}(f_m(x_{m-}^H, u_m^H)) \leq Q_m^{H+1}(f_m(x_{m-}^H, u_m^H)) + (d_{\mathcal{L}} - d + 1)\varepsilon.$$

We show that for every node n with $d_n = d - 1$

$$Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) \leq Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) + (d_{\mathcal{L}} - d + 2)\varepsilon.$$

Since at termination of the algorithm

$$\|x_n^H - f_n(x_{n-}^H, u_n^H)\|_\infty < \delta,$$

it follows that

$$
\begin{aligned}
Q_n^{H+1}(f_n(x_{n-}^H, u_n^H)) &\leq q_n^H && \text{by Lemma 6}\\
&= \sum_{m \in n+} p(m)\left(r_m(x_n^H, \hat{u}_m^H) + Q_m^{H+1}(f_m(x_n^H, \hat{u}_m^H))\right) && \text{by (52)}\\
&= \sum_{m \in n+} p(m)\left(r_m(x_n^H, u_m^H) + Q_m^H(f_m(x_n^H, u_m^H))\right) && \text{by (63)}\\
&\leq \sum_{m \in n+} p(m)\left(r_m(x_n^H, u_m^H) + V_m(f_m(x_n^H, u_m^H) + (d_{\mathcal{L}} - d_m + 1)\varepsilon\right) && \text{induction hypothesis}\\
&\leq \sum_{m \in n+} p(m)\left(r_m(x_n^H, \tilde{u}_m^H) + V_m(f_m(x_n^H, \tilde{u}_m^H))\right) + (d_{\mathcal{L}} - d + 1)\varepsilon && \text{optimality of } \tilde{u}_m^H\\
&= V_n(x_n^H) + (d_{\mathcal{L}} - d + 1)\varepsilon && \text{definition of } V_n\\
&\leq V_n(f_n(x_{n-}^H, u_n^H)) + (d_{\mathcal{L}} - d + 2)\varepsilon
\end{aligned}
$$

where the last inequality follows from $\|x_n^H - f_n(x_{n-}^H, u_n^H)\|_\infty < \delta$ and (49). This establishes the result for nodes with depth $d - 1$ and hence all $n \in \mathcal{N}_*$ by induction. $\qquad\square$

Theorem 4 *Suppose $\bar{\varepsilon} = 0$. Upon termination of the algorithm, the first-stage decisions u_m^H satisfy*

$$\sum_{m \in 0+} p(m)\left(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H))\right) \geq V_0(\bar{x}) - 2(d_{\mathcal{L}} - 1)\varepsilon.$$

Proof For every $m \in 0+$, $d(m) = 2$. So for the optimal first stage decisions u_m^*, Lemma 7 gives

$$r_m(\bar{x}, u_m^*) + Q_m^H(f_m(\bar{x}, u_m^*)) \geq r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon,$$

and by (52)

$$r_m(\bar{x}, u_m^H) + Q_m^H(f_m(\bar{x}, u_m^H)) \geq r_M(\bar{x}, u_m^*) + Q_m^H(f_m(\bar{x}, u_m^*)),$$

so

$$r_m(\bar{x}, u_m^H) + Q_m^H(f_m(\bar{x}, u_m^H)) \geq r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon. \tag{64}$$

Now upon termination of the algorithm (62) gives

$$Q_m^H(f_m(\bar{x}, u_m^H) = Q_m^{H+1}(f_m(\bar{x}, u_m^H), \tag{65}$$

and Lemma 8 implies

$$Q_m^{H+1}(f_m(\bar{x}, u_m^H) \leq V_m(f_m(\bar{x}, u_m^H)) + (d_{\mathcal{L}} - 1)\varepsilon \tag{66}$$

so (64), (65), and (66) yield

$$
\begin{aligned}
&r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) + (d_{\mathcal{L}} - 1)\varepsilon\\
&\geq r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*)) - (d_{\mathcal{L}} - 1)\varepsilon
\end{aligned}
$$

so

$$\sum_{m \in 0+} p(m)(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)))$$

$$\geq \sum_{m \in 0+} p(m)(r_m(\bar{x}, u_m^*) + V_m(f_m(\bar{x}, u_m^*))) - 2(d_{\mathcal{L}} - 1)\varepsilon$$

giving the result. □

4.3 Sampled MIDAS algorithm

We consider next a variant where, in the forward step, only one trajectory is explored, and no stopping test is given.

Algorithm 4 Sampled MIDAS

Set $k = 1$, and $\Omega_n^k = \emptyset$, for all $n \in \mathcal{N}$.

1. Forward pass: Set $x_0^k = \bar{x}$, and $n = 0$. While $n \notin \mathcal{L}$:
   (a) Sample $m \in n+$;
   (b) Solve $\max\limits_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^k(f_m(x_n^k, u)) \right\}$ to give u_m^k;
   (c) If $\left\| f_m(x_n^k, u_m^k) - x_m^h \right\|_\infty < \delta$ for some $h < k$ then set $x_m^k = x_m^h$.
       Otherwise, set $x_m^k = f_m(x_n^k, u_m^k)$.
   (d) Set $n = m$.
2. Leaf node update: Set $\Omega_n^{k+1} := \Omega_n^k$, for all $n \in \mathcal{N}$.
   For the particular leaf node $n \in \mathcal{L}$ at the end of step 1:
   (a) Set $q_n^k = R(x_n^k)$ and $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.
3. Backward pass: While $n > 0$:
   (a) Set $n = n-$;
   (b) Compute
   $$q_n^k = \sum_{m \in n+} \frac{p(m)}{p(n)} \left[\max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\} \right]$$
   attained at
   $$\hat{u}_m \in \arg \max_{u \in U_m(x_n^k)} \left\{ r_m(x_n^k, u) + Q_m^{k+1}(f_m(x_n^k, u)) \right\}.$$
   (c) Set $\Omega_n^{k+1} := \Omega_n^{k+1} \cup \{(x_n^k, q_n^k)\}$.
4. Increase k by 1 and go to step 1.

Following [13] we assume that sample paths satisfy the Forward Pass Sampling Property.

Forward Pass Sampling Property (FPSP):

$$\text{Each node is visited infinitely many times with probability 1.} \tag{67}$$

There are many sampling methods satisfying this property. For example, one method is to select a child node at each node n in the forward pass, by independently sampling with a positive probability for each outcome $m \in n+$. This meets FPSP by the Borel-Cantelli lemma. Another sampling method that satisfies FPSP is to repeat an exhaustive enumeration of each scenario in the forward pass.

Recall that $d(n)$ is the depth of node n. The following result ensures that Q_n^k is a $(d_{\mathcal{L}} + 1 - d(n))\varepsilon$-upper bound on V_n. Its proof is identical to the one in the case of the full tree algorithm.

Lemma 9 *For every $n \in \mathcal{N}$, and for all iterations k, (50) holds.*

We can now state a convergence result for MIDAS.

Theorem 5 *Suppose the approximate non-decreasing assumption (49) and FPSP (67) hold. Almost surely, after finitely many (say H) iterations, the functions Q_n^k remain constant. Then, any policy is such that*

$$V_0(\bar{x}) \leq \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n\left(x_{n-}^H, u_n^H\right) + \sum_{n \in \mathcal{L}} p(n) R\left(x_n^H\right) + d_{\mathcal{L}} \varepsilon, \tag{68}$$

and the first-stage decisions satisfy

$$\sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) \right) \geq V_0(\bar{x}) - 2(d_{\mathcal{L}} - 1)\varepsilon. \tag{69}$$

Proof Since the algorithm generates finitely many points, it is clear that the functions Q_n^k remain constant after finitely many iterations. Since the FPSP is satisfied, each node is a.s. visited infinitely many times. We easily see that the estimates of the full tree algorithm are then valid. The conclusion follows. □

In practical implementations of MIDAS, we choose to stop after a fixed number H_{max} of iterations. Since $Q_0^{H_{max}}$ gives a $2(d_{\mathcal{L}} - 1)\varepsilon$-upper bound on any optimal policy, we can estimate an optimality gap by simulating the candidate policy and comparing the resulting estimate of its expected value (and its standard error) with $Q_0^{H_{max}} + 2(d_{\mathcal{L}} - 1)\varepsilon$.

We also remark that Algorithm 4 simplifies when the random variables are stagewise independent. In this case, the points (x_n^k, q_n^k) can be shared across all nodes having the same depth. This means that there is a single approximation Q_n^k shared by all these nodes and updated once for all in each backward pass. The almost-sure convergence result for MIDAS applies in this special case, but one might expect the number of iterations needed to decrease dramatically in comparison with the general case.

4.4 Multistage stochastic integer programming

The MIDAS algorithm described above can be applied to any multistage optimization problem with value functions that can be approximated by monotonic piecewise constant functions. The accuracy of the approximation relies on the approximate non-decreasing assumption (49), namely

$$V_n(x) \leq V_n(y) + \varepsilon, \quad \text{if } x \leq y + \delta\mathbf{1}, \quad \text{for all } x, y \text{ in } X_n. \tag{70}$$

We can apply the algorithm to a problem in which x is required to be integer so $V_n(x)$ is defined only at integer points. In the formulation MSPT, define $U_n(x_{n-}) \subseteq \mathbb{Z}^M$, and define dynamics so that $X_n \subseteq \mathbb{Z}^N$. Now observe that if $V_n(x)$ is known to be nondecreasing in x and we choose $\delta < 1$ then (70) holds with $\varepsilon = 0$. We then have that MIDAS converges almost surely to an optimal solution in a finite number of iterations.

Theorem 6 *Suppose MIDAS with $\delta \in (0, 1)$ and FPSP is applied to a problem in which x is required to be integer and the Bellman functions $V_n(x)$, $n \in \mathcal{N}_*$ are known to be nondecreasing in x. Almost surely, after finitely many (say H) iterations, the functions Q_n^k remain constant. Then, any policy is such that*

$$V_0(\bar{x}) \leq \sum_{n \in \mathcal{N}_* \setminus \mathcal{L}} p(n) r_n\left(x_{n-}^H, u_n^H\right) + \sum_{n \in \mathcal{L}} p(n) R\left(x_n^H\right). \tag{71}$$

and the first-stage decisions satisfy

$$\sum_{m \in 0+} p(m) \left(r_m(\bar{x}, u_m^H) + V_m(f_m(\bar{x}, u_m^H)) \right) \geq V_0(\bar{x}). \tag{72}$$

Proof If x is integer then $V_n(x)$ jumps only at integer points. Since $\delta < 1$ and $V_n(x)$ is nondecreasing in x, the approximate non-decreasing assumption (49) is true with $\varepsilon = 0$. Since, in addition, FPSP holds, Theorem 5 yields termination after H iterations almost surely and (68) and (69) with $\varepsilon = 0$. This gives (71) and (72) as required. □

In practical implementations of MIDAS applied to multistage stochastic integer programs, the choice of $\delta < 1$ might lead to only incremental changes in u_m at each iteration, resulting in slow convergence. It is possible that choosing δ larger than 1 would lead to faster convergence to good policies, albeit with a weaker bound on their distance from optimality. In the next section we illustrate this phenomenon on a simple example.

5 Numerical examples

To illustrate MIDAS we apply it to an example problem with a continuous state variable. This problem is an instance of a single-reservoir hydroelectric scheduling problem, as described in section 1. In the model we consider, the state $x = (s_t, p_t)$ represents both a reservoir stock variable s_t and a price variable p_t with the following dynamics

$$\begin{bmatrix} s_{t+1} \\ p_{t+1} \end{bmatrix} = \begin{bmatrix} s_t - v_t - l_t + \omega_t \\ \alpha_t p_t + (1 - \alpha_t)b_t + \eta_t \end{bmatrix},$$

where v_t is reservoir release, l_t is reservoir spill, ω_t is (random) reservoir inflow, and η_t is the error term for an autoregressive model of price that reverts to a mean price b_t. This means that $\xi_t = [\,\omega_t\ \eta_t\,]^\top$. We define the reward function as the revenue earned by the released energy $g(v)$ sold at price p,

$$r_t(s, p, v, \omega_t, \eta_t) = pg(v).$$

We approximate the value function by piecewise constant functions in two dimensions, where q^h represents the expected value function estimate at s^h and p^h. Following the formulation in the Appendix, the approximate value function $Q_t^k(s_t, p_t)$ for storage s_t and price p_t is defined as:

$$Q_t^k(s_t, p_t) = \max \quad \varphi \tag{73}$$
$$\text{s.t.}$$
$$\begin{aligned}
\varphi &\leq q_t^h + (\bar{V} - q_t^h)(1 - w_h), \quad h = 1, \ldots, H, \\
s_t &\geq s_t^h z_s^h + \delta_s, \quad h = 1, \ldots, H, \\
p_t &\geq p_t^h z_p^h + \delta_p, \quad h = 1, \ldots, H, \\
z_s^h + z_p^h &= 1 - w_h, \quad h = 1, \ldots, H, \\
w_h, z_s^h, z_p^h &\in \{0, 1\}, \quad h = 1, \ldots, H.
\end{aligned}$$

Figure 3 illustrates an example of how the value function is approximated. At $(s_t, p_t) = (175, 17.5)$, the expected value function estimate is 4000.
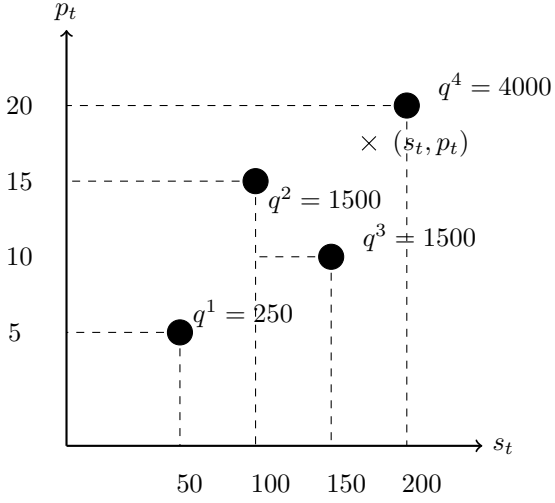
Fig. 3 Value function approximation for $k = 4$. $Q_t^k(x)$ at the point $x = (s_t, p_t)$ shown by the cross equals $q^4 = 4000$.

We applied Algorithm 4 to problems ranging from $T = 5$ to $T = 12$ stages. The reservoir has capacity 100. The generation function $g(v)$ is a convex piecewise linear function of the turbine flow v, which is at most 70 units unless the reservoir contents are smaller than this. So the stage problem (apart from the Bellman function) can be modeled as a linear program. Table 1 summarizes the values of the parameters chosen.

Parameter	Value
T	$5, \ldots, 12$
α_t for $t = 1, 2, \ldots, T$	0.5
η_t for $t = 1, 2, \ldots, T$	Normal$(0, 1)$
b_t	[61.261, 56.716, 59.159, 66.080, 72.131, 76.708, 76.665, 76.071, 76.832, 69.970, 69.132, 67.176]
X_δ	$[\delta_s, 100] \times [\delta_p, 85]$
l for $t = 1, 2, \ldots, T$	0
ω_t for $t = 1, 2, \ldots, T$	0
$U(s_t)$	$v \in [[0, 70] \cap [0, s_t]$
$g(v)$	$\begin{cases} 1.1v & \text{if } v \in [0, 50], \\ v + 5 & \text{if } v \in (50, 60], \\ 0.5(v - 60) + 65 & \text{if } v \in (60, 70], \\ 0 & \text{otherwise.} \end{cases}$
Initial s	100
Initial p	61.261

Table 1 Model parameters for single-reservoir, twelve-stage hydroscheduling problem.

The price was modelled as an autoregressive lag 1 process that reverts to a mean defined by b_t where the noise term η_t is assumed to have a standard normal distribution that is discretized into 5 outcomes. The approximated price process is illustrated by Figure 4.
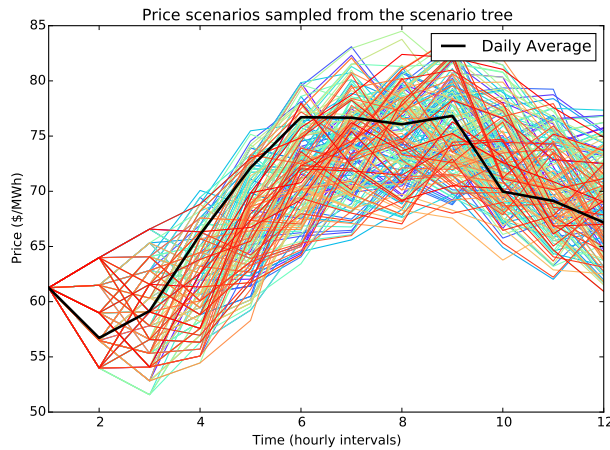
Fig. 4 Sampled price scenarios from a 12 stage price process with 5 discrete η_t values. The full tree has 48,828,125 scenarios.

MIDAS was coded in Python on a *Dell XPS 13* with an Intel Core i5 processor and 8GB of RAM. All MIP problems in MIDAS were solved using CPLEX 12.6 under default settings. The first set of results illustrates the performance of MIDAS with $\delta_s = \delta_p = 1$ on problems with an increasing number of stages. MIDAS was run for 20 hours on each problem instance, and the $2T\varepsilon$-upper bounds recorded at each iteration. These are plotted in Figure 5. The bounds decrease and converge after about 1500 iterations. Occasionally a big decrease in bound is obtained. This occurs when MIDAS adds a new point x_n^k to Ω_n^k that is some distance away from the points already in Ω_n^k. In most of the other iterations the new state added is at ℓ_∞ distance $\delta = 1$ from the set Ω_n^k, so the upper bound decreases slowly.
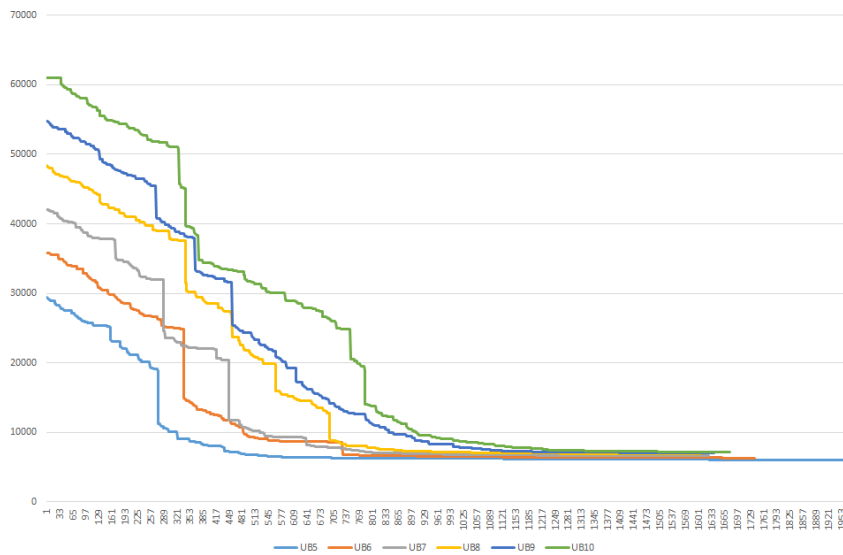

Fig. 5 $2T\varepsilon$-upper bounds of MIDAS for different stage lengths. Here UBT denotes the upper bound for a T-stage problem.

For comparison, we constructed the deterministic equivalent linear programming problem corresponding to the full scenario tree and solved this on a 20 CPU virtual machine with 64 GB of RAM using Gurobi 7.5.2. The results for $T = 5$ to $T = 10$ are shown in Table 2.

	Deterministic equivalent tree					MIDAS ($\delta_s = 1$, $\delta_p = 1$)			
Stages	Nodes	Rows	Columns	Objective	Time (sec)	Lower bound	Standard error	$2\varepsilon T$-Upper bound	Upper bound
5	781	19530	31248	6593.14	0.27	6194.30	141.02	6089.45	7939.45
6	3906	97655	156248	7062.79	1.26	6532.16	226.77	6332.73	8552.73
7	19531	488280	781248	7365.29	6.16	6782.94	266.79	6622.91	9212.91
8	97656	2441405	3906248	7495.75	32.60	7010.64	313.75	6876.08	9836.08
9	488281	12207030	19531248	7579.33	197.41	7038.11	258.76	7050.20	10380.20
10	2441406	61035155	97656248	-	-	7101.84	247.58	7164.96	10864.96

Table 2 MIDAS performance with increasing number of stages

For instances with 9 or fewer stages the deterministic equivalent tree formulation solves very fast to give the true optimal objective value for these models. However, as the number of stages increases the size of the model grows exponentially, so when applied to the instance with 10 stages, Gurobi runs out of memory on the 64GB virtual machine. Similar memory limits were encountered earlier (at 8 stages) on the 8GB machine.

MIDAS gives solutions after 20 hours of computation that have $2\varepsilon T$-upper bounds that are below the true optimal value. We can estimate ε for this single reservoir problem as $185 = 85\delta_s + 100\delta_p$, since the maximum price over the planning horizon is no more than 85 (see Figure 4), and the maximum we can store is 100, since we assume zero inflows. This results in adjusted values in the column denoted "Upper bound" in Table 2.

Simulation of the MIDAS policies obtained in each instance (with 100 simulations) gives estimated revenues shown in the column "Lower bound" in Table 2. These have standard errors of the order of 250 as shown, so 95% confidence intervals of the MIDAS policy values are quite wide. In all cases these do not cover the upper bound, so we cannot claim that the policies are close to optimal. The true optimal policies have objectives shown in the fifth column of Table 2. These are reasonably close to the upper 95% confidence bound of the MIDAS policies (although we could not tell this without the true optimal values).

Figures 6 and 7 show the results from applying MIDAS to the same problem with 12 stages, setting $\delta_p = 5$ and choosing various δ_s values (denoted δ_x in the figures). For $\delta_x = 50$, one can see from Figure 6 that the $2\varepsilon T$-upper bound (shown in red) converges rapidly to a low value. Recall that this choice of δ_x results in ε being potentially as large as $4750 = 85 * 50 + 100 * 5$, which provides little confidence in the solution. For $\delta_x = 10$, the $2\varepsilon T$-upper bound (shown in purple) converges more slowly. Here $\varepsilon \leq 1350$, so one might expect a more profitable policy. This is confirmed in Figure 7, which shows the results of simulating the policies obtained at each iteration of MIDAS using different δ_x values.
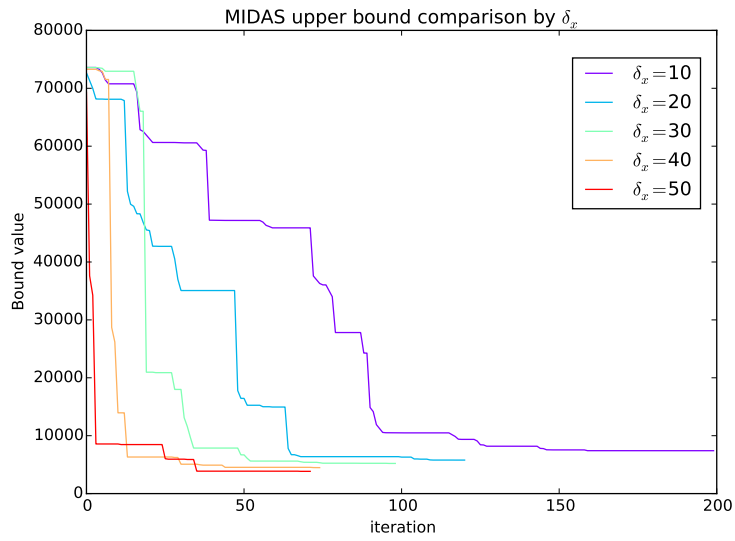
Fig. 6 Comparison of upper bound for various δ_x values of the MIDAS algorithm for a single reservoir hydro scheme.
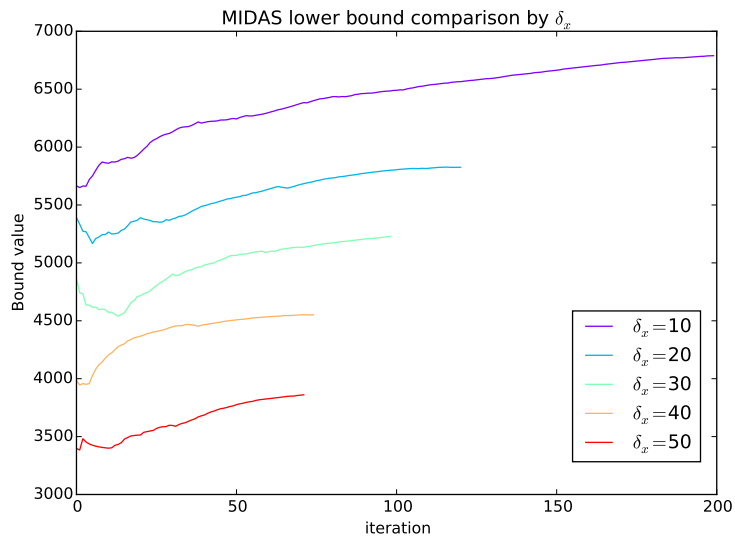

Fig. 7 Comparison of lower bound for various δ_x values of the MIDAS algorithm for a single reservoir hydro scheme.

Finally in Figure 8 we show the storage trajectories that are obtained in the forward pass of MIDAS when applied to the 12-stage problem with $\delta_s = 10$ and $\delta_p = 5$. One can see from this figure that the algorithm visits high storage states more often in periods 1 through 8, as the policy converges towards one that will retain water for the highest price periods 6 to 10 (see Figure 4).
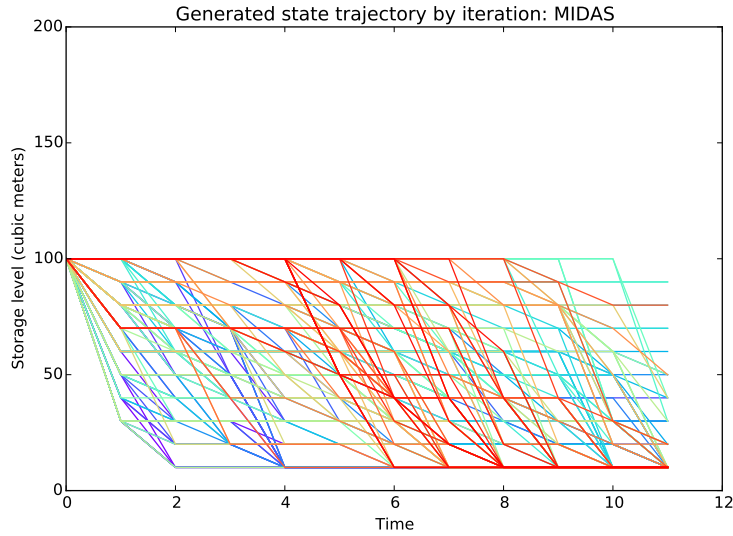
Fig. 8 Generated state trajectories for each iteration of the MIDAS algorithm ($\delta_s = 10$). The coloring shows how many simulations visit the state, where red indicates many visits, and blue infrequent visits.

6 Conclusion

In this paper, we have proposed a method called MIDAS for solving multistage stochastic dynamic programming problems with nondecreasing Bellman functions. For a problem with T stages, we have demonstrated the almost-sure convergence of MIDAS to a $2T\varepsilon$-optimal policy in a finite number of steps. Based on the numerical example in section 5, we may expect that the number of iterations will decrease as δ increases. Increasing δ increases the distance between distinct states that are visited, hence reducing the number of piecewise constant functions needed to cover the domain of the value function. However, the quality of the optimal policy will depend on the value of δ, since a smaller δ will give a lower ε. Furthermore, since the current formulation treats each stage problem as a MIP, the stage problems will increase in size with each iteration. Therefore it is important to choose an appropriate δ to produce good optimum policies within reasonable computation time.

MIDAS can be extended in several ways. Currently, we approximate the stage problems using a MIP formulation that treats the Bellman functions as piecewise constant. One formulation of such a MIP is given in the Appendix. A more accurate approximation might be achieved using a stronger MIP formulation that combines piecewise constant functions with cutting planes. As long as this provides an ε-outer approximation of the Bellman function, we can incorporate it into a MIDAS scheme, which will converge almost surely by the same arguments above.

The application motivating this paper yields nonconcave Bellman functions by virtue of the mean-reverting price process. MIDAS can compute approximate solutions to instances of this problem that defeat state-of-the-art LP solvers when applied to a scenario-tree version of the model.

Although our theoretical analysis has focused on problems with continuous Bellman functions, MIDAS can be applied to multistage stochastic integer programming (MSIP) problems. Indeed as Theorem 6 demonstrates, MIDAS with $\delta < 1$ will converge almost surely to an optimal solution to a MSIP. Solving a deterministic equivalent of a MSIP can be difficult, as the scenario tree grows exponentially with the number of stages and the number of outcomes, and MIP algorithms generally scale poorly. However, since solving many small MIPs will be faster than solving a single large MIP, we might expect MIDAS to produce good candidate solutions to

MSIP problems with less computational effort. Our hope is that MIDAS will provide a practical computational approach to solving these difficult multistage stochastic integer programming problems.

Appendix: A MIP representation of $Q^{k+1}(x)$

Assume that $X = \{x : 0 \leq x_i \leq K_i, i = 1, 2, \ldots, N\}$, and let $V(x)$ be any upper semi-continuous function defined on X. Suppose for some points x^h, $h = 1, 2, \ldots, k$, we have $V(x^h) = q^h$. Recall $\bar{V} = \max_{x \in X} V(x)$,

$$\mathcal{H}^{k+1}(x) = \{1 \leq h \leq k : x_i^h > x_i - \delta, i = 1, 2, \ldots, N\},$$

$$Q^{k+1}(x) = \min\left\{\bar{V}, \min\left\{q^h : h \in \mathcal{H}^{k+1}(x)\right\}\right\}.$$

For $\delta > 0$, and for any $x \in X_\delta = \{x : \delta \leq x_i \leq K_i, i = 1, 2, \ldots, n\}$, define $\bar{Q}^{k+1}(x)$ to be the optimal value of the mixed integer program

$$
\begin{aligned}
\text{MIP}(x): \max \quad & \varphi \\
\text{s.t.} \quad & \varphi \leq q^h + (\bar{V} - q^h)(1 - w_h), && h = 1, 2, \ldots, k, \\
& x_i \geq x_i^h z_i^h + \delta, && i = 1, 2, \ldots, N, \\
& \sum_{i=1}^n z_i^h = 1 - w_h, && h = 1, 2, \ldots, k, \\
& w_h \in \{0, 1\}, && h = 1, 2, \ldots, k, \\
& z_i^h \in \{0, 1\}, && i = 1, 2, \ldots, N, \\
& && h = 1, 2, \ldots, k.
\end{aligned}
$$

Proposition 1 *For every $x \in X_\delta$,*

$$\bar{Q}^{k+1}(x) = Q^{k+1}(x).$$

Proof For a given point $x \in X_\delta$, consider w_h, z_i^h, $i = 1, 2, \ldots, N$, $h = 1, 2, \ldots, k$ that are feasible for MIP(x). If $w_h = 0$, $h = 1, 2, \ldots, k$, then $\varphi \leq \bar{V}$ is the only constraint on φ and so $\bar{Q}^k(x) = \bar{V}$. But $w_h = 0$, $h = 1, 2, \ldots, k$ means that for every such h, $z_i^h = 1$ for some component i giving

$$x_i \geq x_i^h + \delta.$$

Thus

$$\mathcal{H}^{k+1}(x) = \{h : x_i < x_i^h + \delta \text{ for every } i\} = \emptyset.$$

Thus $Q^{k+1}(x) = \bar{V}$ which is the same value as $\bar{Q}^{k+1}(x)$.

Now assume that the optimal solution to MIP(x) has $w_h = 1$ for some h. It suffices to show that

$$\bar{Q}^{k+1}(x) = \min\{q^h : h \in \mathcal{H}^{k+1}(x)\}.$$

First if $h \in \mathcal{H}^{k+1}(x)$ then $w_h = 1$. This is because choosing $w_h = 0$ implies $z_i^h = 1$ for some i, so for at least one i

$$x_i \geq x_i^h + \delta,$$

so $h \notin \mathcal{H}^{k+1}(x)$.

Now if $h \notin \mathcal{H}^{k+1}(x)$ then any feasible solution to MIP(x) can have either $w_h = 0$ or $w_h = 1$. Observe however that if

$$q^h < \min\{q^{h'} : h' \in \mathcal{H}^{k+1}(x)\}$$

for any such h then choosing $w_h = 1$ for any of these would yield a value of φ strictly lower than the value obtained by choosing $w_h = 0$ for all of them. So $w_h = 0$ is optimal for $h \notin \mathcal{H}^{k+1}(x)$. It follows that $\mathcal{H}^{k+1}(x) = \{h : w_h = 1\}$. Thus the optimal value of MIP(x) is

$$\bar{Q}^{k+1}(x) = \min\{q^h : w_h = 1\} = \min\{q^h : h \in \mathcal{H}^{k+1}(x)\} = Q^{k+1}(x).$$

$\square$

References

1. H. Abgottspon, K. Njalsson, M.A. Bucher, and G. Andersson. Risk-averse medium-term hydro optimization considering provision of spinning reserves. In *2014 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6. IEEE, 2014.
2. J.F. Bonnans, Z. Cen, and Th. Christel. Energy contracts management by stochastic programming techniques. *Ann. Oper. Res.*, 200:199–222, 2012.
3. S. Cerisola, J.M. Latorre, and A. Ramos. Stochastic dual dynamic programming applied to nonconvex hydrothermal models. *European Journal of Operational Research*, 218(3):687–697, 2012.
4. Z. Chen and W.B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.
5. C.J. Donohue and J.R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1(1), 2006.
6. A. Downward, O. Dowson, and R. Baucke. Stochastic dual dynamic programming with stagewise dependent objective uncertainty. *Available on Optimization Online*, 2018.
7. P. Girardeau, V. Leclere, and A.B. Philpott. On the convergence of decomposition methods for multistage stochastic convex programs. *Mathematics of Operations Research*, 40(1):130–145, 2014.
8. A. Gjelsvik, B. Mo, and A. Haugstad. Long-and medium-term operations planning and stochastic modelling in hydro-dominated power systems based on stochastic dual dynamic programming. In *Handbook of power systems I*, pages 33–55. Springer, 2010.
9. V. Guigues. Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs. *SIAM Journal on Optimization*, 26(4):2468–2494, 2016.
10. M. Hindsberger and A.B. Philpott. Resa: A method for solving multistage stochastic linear programs. *Journal of Applied Operational Research*, 6(1):2–15, 2014.
11. M.N. Hjelmeland, J. Zou, A. Helseth, and S. Ahmed. Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. *IEEE Transactions on Sustainable Energy*, 2018.
12. M.V.F. Pereira and L.M.V.G. Pinto. Multistage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991.
13. A.B. Philpott and Z. Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455, 2008.
14. A. Shapiro. On complexity of multistage stochastic programs. *Operations Research Letters*, 34(1):1 – 8, 2006.
15. A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.
16. G. Steeger and S. Rebennack. Strategic bidding for multiple price-maker hydroelectric producers. *IIE Transactions*, 47(9):1013–1031, 2015.
17. F. Thome, M.V.F. Pereira, S. Granville, and M. Fampa. Non-convexities representation on hydrothermal operation planning using SDDP. Technical report, working paper, available: http://www. psr-inc. com. br/portal/psr/publicacoes, 2013.
18. F. Wahid. *River optimization: short-term hydro biding under uncertainty*. PhD thesis, University of Auckland/Ecole Polytechnique, 2017.
19. J. Zou, S. Ahmed, and X. A. Sun. Nested decomposition of multistage stochastic integer programs with binary state variables. *Available on Optimization Online*, 2016.