algorithms

Article

An Algorithm for Making Regime-Changing Markov Decisions

Juri Hinz

School of Mathematical and Physical Sciences, University of Technology Sydney, P.O. Box 123, Ultimo, NSW 2007, Australia; juri.hinz@uts.edu.au

Abstract: In industrial applications, the processes of optimal sequential decision making are naturally formulated and optimized within a standard setting of Markov decision theory. In practice, however, decisions must be made under incomplete and uncertain information about parameters and transition probabilities. This situation occurs when a system may suffer a regime switch changing not only the transition probabilities but also the control costs. After such an event, the effect of the actions may turn to the opposite, meaning that all strategies must be revised. Due to practical importance of this problem, a variety of methods has been suggested, ranging from incorporating regime switches into Markov dynamics to numerous concepts addressing model uncertainty. In this work, we suggest a pragmatic and practical approach using a natural re-formulation of this problem as a so-called convex switching system, we make efficient numerical algorithms applicable.

Keywords: Markov decisions; partial observability

check for updates

Citation: Hinz, J. An Algorithm for Making Regime-Changing Markov Decisions. *Algorithms* **2021**, *14*, 291. https://doi.org/10.3390/a14100291

Academic Editor: Mihaly Mezei

Received: 23 August 2021
Accepted: 30 September 2021
Published: 4 October 2021

1. Introduction

Decision-theoretic planning is naturally formulated and solved using Markov Decision Processes (MDPs, see [1]). This theory provides a fundamental and intuitive formalism not only for sequential decision optimization, but also for diverse learning problems in stochastic domains. A typical goal in such framework is to model an environment as a set of states and actions that can be performed to control these states. Thereby, the goal is to drive the system maximizing specific performance criteria.

The methodologies of MDPs have been successfully applied to (stochastic) planning, learning, robot control, and game playing problems. In fact, MDPs nowadays provide a standard toolbox for learning techniques in sequential decision making. To explain our contribution to this traditional and widespread area, let us consider a simplified example of motion control. Suppose that a robot is moving in two horizontal directions on a rectangular grid whose cells are identified with the states of the system. At any time step, there are four actions to guide the robot from the current position to one of the neighboring cells. These actions are UP, DOWN, RIGHT, LEFT, which command a move to the corresponding direction. However, the success of these actions is uncertain and is state-dependent: For instance, a command (UP) may not always cause a transition to the intended (upper) cell, particularly if the robot is at the (upper) boundary. The controller aims to reach a pre-specified target cell at minimal total costs. These costs are accumulated each time when a control is applied and depend on the current position—the cell currently occupied—accounting for obstacles and other adverse circumstances that may be encountered at some locations. In mathematical terms, such a motion is determined by a so-called controlled discrete-time discrete-space Markov chain. In a more realistic situation, the environment is dynamic: The target can suddenly change its location, or navigation through certain cells can become more difficult, changing the control costs and transitions. In principle, such problems can be addressed in terms of the so-called *partially observable Markov decision processes* (POMDPs, see [2]), but this approach may turn out to be cumbersome due its higher complexity than that of ordinary MDPs. Instead, we suggest a technique which overcomes this difficulty by a natural and direct modeling in terms of a finite number of Markov decision processes (sharing the same set of sets and actions), each active in a specific regime, when the regime changes, another Markov decision processes takes over. Thereby, the regime is not directly observable, thus the controller must guess which of those Markov decision processes is valid at the current decision time: Determining an optimal control becomes challenging due to regime switches. Surprisingly, one can re-formulate such control problems as a *convex switching system* [3], in order to take advantage from efficient numerical schemes and advanced error control. As a result, we obtain sound algorithms for solutions of regime-modulated Markov decision problems.

This work models random environment as selection of a finite-number ordinary Markov Decision Processes which are mixed by uncertain observations. On this account, we follow a traditional path facing well-known difficulties originated from high-dimensional state spaces and all problems from incorporating an information flow into a centralized decision process. However, let us emphasize that there is an approach which aims to bypass some of the problems through an alternative modeling. Namely, a game-theoretic framework has attracted significant attention (see [4,5]) as an alternative to the traditional centralized decision optimization within a random and dynamic environment. Here, individual agents aiming at selfishly maximizing their own wealth are acting in a game whose equilibrium replaces a centralized decision optimization. In such a context, the process of gathering and utilizing information is easier to model and manage since the individual strategy optimization is simpler and more efficient in collecting and processing all relevant information which may be private, noisy and highly dispersed.

Before we turn to technical details, let us we summarize in the notations and abbreviations used in this work Table 1.

Table 1. Notations and measurement units.

stochastic kernel	$\mathcal{K}_t^a$	vector with unit entries	$\vec{1}$	volume	cubic meter
controlled probability	$\mathbb{P}^{x,\pi}$	l^1-norm	$\|\cdot\|$	area	hectare
controlled expectation	$\mathbb{E}^{x,\pi}$	maximum	$\bigvee$	currency	USD
Bellman operator	$\mathcal{T}_t$	binding by row	$\sqcup$	costs	USD per hectare

2. Discrete-Time Stochastic Control

First, let us review a finite-horizon control theory. Consider a random dynamics within a time horizon $\{0, 1, \ldots, T\} \subset \mathbb{N}$ whose state x evolves in a state space $\mathcal{X}$ (subset of a Euclidean space) and is controlled by actions a from a finite action set A. The mapping $\pi_t : \mathcal{X} \mapsto A$, describing the action $\pi_t(x)$ that the controller takes at time $t = 0, \ldots T$ in the situation $x \in \mathcal{X}$ is referred to as *decision rule*. A sequence of decision rules $\pi = (\pi_t)_{t=0}^{T-1}$ is called a *policy*. Given a family of

$$\text{stochastic kernels } \mathcal{K}_t^a(x, dx'), \quad t = 0, \ldots, T-1 \;\; a \in A \text{ on } \mathcal{X},$$

there exists an appropriately constructed probability space which supports a stochastic process $(X_t)_{t=0}^{T}$ such that for each initial point $x_0 \in \mathcal{X}$ and each policy $\pi = (\pi_t)_{t=0}^{T-1}$ there exists a measure $\mathbb{P}^{x_0,\pi}$ and such that

$$\mathbb{P}^{x_0,\pi}(X_0 = x_0) = 1, \quad \mathbb{P}^{x_0,\pi}(X_{t+1} \in B \,|\, X_0, \ldots, X_t) = \mathcal{K}_t^{\pi_t(X_t)}(X_t, B) \tag{1}$$

holds for $t = 0, \ldots, T-1$ for each measurable $B \subset \mathcal{X}$. Such a system is called *controlled Markovian evolution*. The interpretation is that if at time t the process state is X_t and the action $\pi_t(X_t)$ is applied, then the distribution $K_t^{\pi_t(X_t)}(X_t, \cdot)$ randomly changes the state from X_t to X_{t+1}.

Stochastic kernels are equivalently described in terms of transition operators. The transition operator, associated with the stochastic kernel $\mathcal{K}_t^a$ will be denoted by the same letter, but it acts on functions $v : \mathcal{X} \to \mathbb{R}$ by

$$(\mathcal{K}_t^a vs.)(x) = \int_{\mathcal{X}} v(x')\mathcal{K}_t^a(x, dx') \qquad x \in \mathcal{X},\, t = 0, \ldots, T-1 \tag{2}$$

whenever the above integrals are well-defined.

Having introduced such controlled Markovian dynamics, let us turn to control costs now. Suppose that for each time $t = 0, \dots, T-1$, we are given the *t-step reward function* $r_t : \mathcal{X} \times A \mapsto \mathbb{R}$ where $r_t(x, a)$ represents the reward for applying an action $a \in A$ when the state of the system is $x \in \mathcal{X}$ at time t. At the end of the time horizon, at time T, it is assumed that no action can be taken. Here, if the system is in a state x, a *scrap value* $r_T(x)$, described by a pre-specified *scrap function* $r_T : \mathcal{X} \to \mathbb{R}$ is collected. The expectation of the cumulative reward from following a policy π is referred to as *policy value*:

$$v_0^\pi(x) = \mathbb{E}^{x,\pi}\left(\sum_{t=0}^{T-1} r_t(X_t, \pi_t(X_t)) + r_T(X_T)\right), \quad x \in \mathcal{X},$$

where $\mathbb{E}^{x,\pi}$ denotes the expectation over the controlled Markov chain whose distribution is defined by (1). For $t = 0, \dots, T-1$, introduce the backward induction operator

$$\mathcal{T}_t^a v(x) = r_t(x, a) + \mathcal{K}_t^a v(x), \quad x \in \mathcal{X}, \quad a \in A \tag{3}$$

which acts on each measurable function $v : \mathcal{X} \to \mathbb{R}$ where $\mathcal{K}_t^a v$ is well-defined. The policy value is obtained as a result of the recursive procedure

$$v_T^\pi = r_T, \quad v_t^\pi(x) = \mathcal{T}_t^{\pi(x)} v_{t+1}^\pi(x) \qquad x \in \mathcal{X}, \text{ for } t = T-1, \dots, 0 \tag{4}$$

which is referred to as the *policy iteration*. The value of a "best possible" policy is addressed using

$$v_0^*(x) = \sup_\pi v_0^\pi(x), \quad x \in \mathcal{X}, \tag{5}$$

where π runs through the set of all policies, the function v_0^* is called the *value function*. The goal of the optimal control is to find a policy $\pi^* = (\pi_t^*)_{t=0}^{T-1}$ where the above maximization is attained:

$$v_0^{\pi^*}(x) = v_0^*(x) \quad \text{for each } x \in \mathcal{X}.$$

Such policy optimization is well-defined (see [6], p. 199). Thereby, the calculation of an optimal policy π^* is performed in the following framework: For $t = 0, \dots, T-1$, introduce the *Bellman operator*

$$\mathcal{T}_t v(x) = \max_{a \in A}(r_t(x, a) + \mathcal{K}_t^a v(x)), \quad x \in \mathcal{X} \tag{6}$$

which acts on each measurable function $v : \mathcal{X} \to \mathbb{R}$ where $\mathcal{K}_t^a v$ is well-defined for all $a \in A$. Further, consider the *Bellman recursion*, (also called backward induction)

$$v_T^* = r_T, \quad v_t^* = \mathcal{T}_t v_{t+1}^* \qquad \text{for } t = T-1, \dots, 0. \tag{7}$$

There solution $(v_t^*)_{t=0}^T$ to the above Bellman recursion returns the *value function* v_0^* and determines an optimal policy π^* via

$$\pi_t^*(x) = \text{argmax}_{a \in A}\big(r_t(x, a) + \mathcal{K}_t^a v_{t+1}^*(x)\big), \quad x \in \mathcal{X}, \quad t = 0, \dots, T-1. \tag{8}$$

Remark 1. *In practice, discounted versions the above stochastic control are popular. These are obtained by replacing the stochastic kernel $\mathcal{K}_t^a$ by $\kappa \mathcal{K}_t^a$ in the backward induction where $\kappa \in [0, 1]$ is a discount factor. The advantage of this approach is that for long time horizons, the optimal policy becomes stationary, provided that all rewards and transition kernels are time-independent.*

3. Markov Decisions under Partial Observation

In view of the general framework from Section 2, the classical Markov Decision Processes (MDPs) are obtained by specifying controlled Markovian evolution $(X_t)_{t=0}^T$ in terms of assumptions on the state space and on the transition kernels. Here, the state space is given by a finite set P such that states are driven in terms of

$$\text{stochastic matrices } (\alpha^a_{p,p'})_{p,p' \in P}, \quad a \in A. \tag{9}$$

indexed by a finite number of actions $a \in A$ with the interpretation that $\alpha^a_{p,p'} \in [0,1]$ stands for the transition probability from $p \in P$ to $p' \in P$ if the action $a \in A$ was taken. In this settings, the stochastic kernels $\mathcal{K}^a_t$ are acting functions $v : P \to \mathbb{R}$ by

$$\mathcal{K}^a_t v(p) = \sum_{p' \in P} \alpha^a_{p,p'} v(p') \qquad p \in P,\ t = 0, \dots T-1. \tag{10}$$

There is no specific assumption on scrap and reward functions, they are given by following mappings on the state space P:

$$p \to r_T(p), \quad (p,a) \mapsto r_t(p,a), \quad t = 0, \dots, T-1, \quad a \in A. \tag{11}$$

In this work, we consider control problems where a finite number of Markov decision problems are involved, sharing the same state space: each is activated by a certain regime. For this reason, we introduce a selection of MDPs indexed by a finite set S of regimes. Here, we assume that stochastic matrices as in (9) and control costs as in (11) are now indexed by S:

$$\begin{matrix} (\alpha^a_{p,p'}(s))_{p,p' \in P}, & p \to r_T(p)(s), \\ (p,a) \mapsto r_t(p,a)(s), & t = 0, \dots, T-1, \end{matrix} \quad a \in A, \qquad s \in S. \tag{12}$$

Let us now consider decision making under incomplete information. Given family of Markov decision problems as in (12), the controller deals with a dynamic mixture of these problems in the sense that each of these MDPs becomes valid under a certain regime which changes exogenously in an uncontrolled way and is not observed directly. More precisely, interpreting each probability distribution $\hat{s} = (\hat{s}(s))_{s \in S}$ on S as controllers believe about the current regime, we introduce the following convex mixtures of the ingredients (12):

$$\begin{matrix} \alpha^a_{p,p'}(\hat{s}) = (\sum_{s \in S} \hat{s}(s) \alpha^a_{p,p'}(s))_{p,p' \in P} & p \to \sum_{s \in S} \hat{s}(s) r_T(p)(s), \\ (p,a) \mapsto \sum_{s \in S} \hat{s}(s) r_t(p,a)(s), & t = 0, \dots, T-1, \end{matrix} \quad a \in A,\ \hat{s} \in \widehat{S}. \tag{13}$$

where $\widehat{S}$ stands for the simplex of all probability distributions on S. That is, the transition kernels and the control costs are now modulated by an external variable $\hat{s} \in \widehat{S}$ which represents the evolution of controller's believe about the current situation. To describe its dynamics, we suppose that the information is updated dynamically through the observation of another process $(y_t)_{t=1}^T$ which follows the so-called *hidden Markov dynamics*. Let us introduce this concept before we finish the definition of our regime-changing Markov decision problem.

Consider a time-homogeneous Markov chain $(s_t)_{t=0}^T$ evolving on a

$$\text{finite space } S \text{ which is identified with the set of orthonormal basis vectors of a finite-dimensional Euclidean space.} \tag{14}$$

Assume that this Markov chain is governed by the stochastic matrix $\Gamma = (\Gamma_{s,s'})_{s,s' \in S}$. It is supposed that the evolution of $(s_t)_{t=0}^T$ is unobservable, representing a *hidden regime*. The available information is realized by another stochastic process $(y_t)_{t=0}^T$ taking values in a space Y, such that at any time t, the distribution of the next observation y_{t+1} depends on the past $\{y_0, \dots, y_t\}$ through the recent state s_t only. More precisely, we suppose that

$((s_t, y_t))_{t=0}^T$ follows a Markov process whose transition operators are acting on functions $v : S \times Y \to \mathbb{R}$ as

$$(s, y) \mapsto \sum_{s' \in S} \int_Y v(s', y') \Gamma_{s,s'} \mu_s(\mathrm{d}y'). \tag{15}$$

Here, $(\mu_s)_{s\in\mathcal{S}}$ stands for the family of distributions for the next-time observation of y_{t+1}, conditioned on $s_t = s \in S$. For each $s \in S$, we assume that the distribution μ_s is absolutely continuous with respect to a reference measure μ on Y and introduce the densities

$$\nu_s(y) = \frac{\mathrm{d}\mu_s}{\mathrm{d}\mu}(y) \text{ for } y \in Y \text{ and } s \in S.$$

Since the state evolution $(s_t)_{t=0}^T$ is not available, one must rely on the believe distribution $\hat{s}_t$ of the state s_t, conditioned on the observations $y_0, \ldots, y_t$. With this, the *hidden state estimates* $(\hat{s}_t)_{t=0}^T$ yield a process that takes values in the set $\widehat{S}$ of all probability measures on S which is identified with the convex hull of S due to (14). It turns out that although the observation process $(y_t)_{t=0}^T$ is non-Markovian, it can be augmented by the believes process to obtain $(z_t = (\widehat{s}_t, y_t))_{t=0}^T$ which follows a Markov process on the state space $Z = \widehat{S} \times Y$. This process is driven by the transition kernels acting on functions $v : \widehat{S} \times Y \to \mathbb{R}$ as

$$\mathcal{K}_t v(\hat{s}, y) = \int_Y vs.\left(\frac{\Gamma^\top \mathcal{V}(y')\hat{s}}{\|\mathcal{V}(y')\hat{s}\|}, y'\right) \|\mathcal{V}(y')\hat{s}\| \mu(\mathrm{d}y'), \quad (\hat{s}, y) \in \widehat{S} \times Y, \tag{16}$$

for all $t = 0, \ldots T-1$. In this formula, $\mathcal{V}(y)$ stands for the diagonal matrix whose diagonal elements are given by $(\nu_s(y))_{s\in S}$, $y \in Y$ whereas $\| \cdot \|$ represents the usual l_1 norm.

Remark 2. *In the Formula (16) the quantity*

$$\hat{s}(y') := \frac{\Gamma^\top \mathcal{V}(y')\hat{s}}{\|\mathcal{V}(y')\hat{s}\|} \tag{17}$$

represents the updated believe state under the assumption that prior to the observation $y' \in Y$, the believe state was $\hat{s} \in \widehat{S}$. On this account, this vector must be an element of $\widehat{S}$, which is seen as follows (the author would like to thank to anonymous referee for highlighting out this point): Having observed that all entries of $\Gamma^\top \mathcal{V}(y')\hat{s}$ and of $\mathcal{V}(y')\hat{s}$ are non-negative (ensured by multiplication of $\hat{s}$ by matrices with non-negative entries), we have to verify that they sum up to one, thus (17) represents a probability distribution on S. For this, we introduce a vector $\vec{1} = (1, 1, \ldots, 1)$ of ones with our believe state dimension, in order to verify that the scalar product is

$$\vec{1}^\top \hat{s}(y') \quad = \quad \frac{\vec{1}^\top \Gamma^\top \mathcal{V}(y')\hat{s}}{\|\mathcal{V}(y')\hat{s}\|} = \frac{\vec{1}^\top \mathcal{V}(y')\hat{s}}{\|\mathcal{V}(y')\hat{s}\|} = 1$$

in other words, all entries of $\hat{s}(y)$ sum up to one. Here, we have used that $\vec{1}^\top \Gamma^\top = \vec{1}$ since all rows of stochastic matrix Γ are probability vector whose l_1 norm is representable by a scalar product with $\vec{1}$.

The optimal control for such modulated MDP is formulated in terms of the transitions operators, rewards and scrap functions which we define now. With notations introduced above, consider a state space $X = P \times Z$ with $Z = \widehat{S} \times Y$. First, introduce a controlled Markovian dynamics in terms of the transition operators acting on functions $v : P \times Z \to \mathbb{R}$ as

$$\mathcal{K}_t^a v(p, \hat{s}, y) = \sum_{p' \in P} \alpha_{p,p'}^a(\hat{s}) \int_Y v\left(p', \frac{\Gamma^\top \mathcal{V}(y')\hat{s}}{\|\mathcal{V}(y')\hat{s}\|}, y'\right) \|\mathcal{V}(y')\hat{s}\| \mu(\mathrm{d}y') = \mathcal{K}_t^a v(p, \hat{s}) \tag{18}$$

for all $p \in P$, $(\hat{s}, y) \in \widehat{S} \times Y$, $t = 0, \dots T-1$. This kernel describes the following evolution: Based on the current situation $(p, \widehat{s}, y)$, a transition to the next decision state $p' \in P$ occurs according to the mixture $\alpha^a_{p,p'}(\hat{s})$ of transition probabilities introduced in (13). Furthermore, our believe state evolves due to the information update based on the new observations y', as described in (16). Obviously, this transformation does not depend on the decision variable $y \in \mathcal{Y}$, with abbreviation introduced by the last equality of (18). Finally, we introduce the control costs that are expressed by scrap and reward functions in terms of mixtures (13):

$$r_T(p, \hat{s}, y) = \sum_{s \in S} \hat{s}(s) r_T(p)(s) = r_T(p, \hat{s}), \tag{19}$$

$$r_t(p, \hat{s}, y, a) = \sum_{s \in S} \hat{s}(s) r_t(p, a)(s) = r_t(p, \hat{s}, a), \tag{20}$$

for $t = 0, \dots, T-1$, $a \in A$, $\hat{s} \in \widehat{S}$, and $y \in Y$. Note that (18), (20) and (19) uniquely define a sequential decision problem in terms of specific instances to controlled dynamics and its control costs. In what follows, we show that this problem seamlessly falls under the umbrella of a general scheme that allows an efficient numerical treatment.

4. Approximate Algorithmic Solutions

Although there are a number theoretical and computational methods to solve stochastic control problems, many industrial applications exhibit complexity and size driving numerical techniques to their computational limits. For an overview on this topic we referrer the reader to [7]. One of the major difficulties originates from high-dimensionality. Here, approximate methods have been proposed based on state and action space discretization or approximations of functions on this space. Among *function approximation* methods, the *least-squares Monte Carlo approach* represents a traditional way to approximate the value function s in [8–13]. However, function approximation methods have also been used to capture local behavior of value functions and advanced regression methods such as kernel methods [14,15], local polynomial regression [16], and neural networks [17], have been suggested.

Considering partial observability, several specific approaches are studied in [18], with bound estimation presented in [19]. The work [20] provides an overview of modern algorithms in this field with the main focus on the so-called point-based solvers. The main aspect of any point-based POMDP algorithm (see [21]) is a dynamical adaptation of the state-discretized grid.

In this work, we treat our regime switching Markov decision problem in terms of efficient numerical schemes, which deliver an approximate solution along with its diagnostics. The following section presents this methodology and elaborates on specific assumptions, required in this setting. The numerical solution method is based on function approximations which require convexity and linear state dynamics. For technical details, we refer the interested reader to [22]. Furthermore, there are applications to pricing financial options [23], natural resource extraction [3], battery management [24] and optimal asset allocation under hidden state dynamics [25], many applications are illustrated using R in [26].

Suppose that the state space $\mathcal{X} = P \times \mathbb{R}^d$ is a Cartesian product of a finite set P and the Euclidean space $\mathbb{R}^d$. Consider a controlled Markovian process $(X_t)_{t=0}^T := (P_t, Z_t)_{t=0}^T$ that consists of two parts. The discrete-space component $(P_t)_{t=0}^T$ describes the evolution of a finite-state controlled Markov chain, taking values in a finite set P, while the continuous-space component $(Z_t)_{t=0}^T$ follows uncontrolled evolution with values in $\mathbb{R}^d$. More specifically, we assume that at any time $t = 0, \dots, T-1$ in an arbitrary state $(p, z) \in \mathcal{X}$ the controller chooses an action a from A in order to trigger the one-step transition from the mode $p \in P$ to the mode $p' \in P$ with probability $\alpha^a_{p,p'}(z)$, given in terms of prespecified transition probability matrices $(\alpha^a_{p,p'}(z))_{p,p' \in P}$ indexed by actions $a \in A$. Note that these transition probabilities can depend on continuous state component $z \in \mathbb{R}^d$. For the continuous-state process $(Z_t)_{t=0}^T$, we assume an uncontrolled evolution which is governed by linear state dynamics

$$Z_{t+1} = W_{t+1} Z_t, \qquad t = 0, \dots, T-1, \tag{21}$$

with independent *disturbance matrices* $(W_t)_{t=1}^T$, thus the transition operators $\mathcal{K}_t^a$ are

$$\mathcal{K}_t^a v(p,z) = \sum_{p' \in P} \alpha_{p,p'}^a(z) \mathbb{E}(v(p', W_{t+1}z)), \quad p \in P, z \in \mathbb{R}^d, t = 0, \dots, T-1, \tag{22}$$

acting on all function $v : P \times \mathbb{R}^d \to \mathbb{R}$ where the required expectations exist. Furthermore, we suppose that the reward and the scrap functions

$$r_t : P \times \mathbb{R}^d \times A \to \mathbb{R}, \;\; r_T : P \times \mathbb{R}^d \to \mathbb{R}, \\ \text{are convex in the second argument.} \tag{23}$$

The numerical treatment aims determining approximations to the true value functions $(v_t^*)_{t=0}^{T-1}$ and to the corresponding optimal policies $\pi^* = (\pi_t^*)_{t=0}^{T-1}$. Under some additional assumptions, the value functions turn out to be convex and can be approximated by piecewise linear and convex functions.

To obtain an efficient (approximative) numerical treatment of these operations, the concept of the so-called *sub-gradient envelopes* was suggested in [22]. A sub-gradient $\triangledown_g f$ of a convex function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $g \in \mathbb{R}^d$ is an affine–linear functional supporting this point $\triangledown_g f(g) = f(g)$ from below $\triangledown_g f \leq f$. Given a finite grid $G = \{g^1, g^2, \dots, g^m\} \subset \mathbb{R}^d$, the sub-gradient envelope $\mathcal{S}_G f$ of f on G is defined as a maximum of its sub-gradients

$$\mathcal{S}_G f = \bigvee_{g \in G} (\triangledown_g f), \tag{24}$$

which provides a convex approximation of the function f from below $\mathcal{S}_G f \leq f$, and enjoys many useful properties. Using the sub-gradient envelope operator, define the double-modified Bellman operator as

$$\mathcal{T}_t^{m,n} v(p,z) = \max_{a \in A} \left(\mathcal{S}_G r_t(p,z,a) + \sum_{p' \in P} \sum_{k=1}^n \nu_{t+1}^{(k)} \mathcal{S}_G \alpha_{p,p'}^a(\cdot) v(p', W_{t+1}^{(k)} \cdot)(z) \right), \tag{25}$$

where the probability weights $(\nu_{t+1}^{(k)})_{k=1}^n$ correspond to the distribution sampling $(W_{t+1}^{(k)})_{k=1}^n$ of each disturbance matrix W_{t+1}. The corresponding backward induction

$$v_T^{m,n}(p,z) = \mathcal{S}_G r_T(p,z), \tag{26}$$
$$v_t^{m,n}(p,z) = \mathcal{T}_t^{m,n} v_{t+1}^{m,n}(p,z), \qquad t = T-1, \dots 0, \tag{27}$$

yields the so-called double-modified value functions $(v_t^{m,n})_{t=0}^T$. Under appropriate assumptions on increasing grid density and disturbance sampling, the double-modified value functions converge uniformly to the true value functions on compact sets (see [22]). The crucial point of our algorithm is a treatment of piecewise linear convex functions *in terms of matrices*. To address this aspect, let us agree on the following notation: Given a function f and a matrix F, we write $f \sim F$ whenever $f(z) = \max(Fz)$ holds for all $z \in \mathbb{R}^d$, and call F *a matrix representative* of f. To be able to capture a sufficiently large family of functions by matrix representatives, an appropriate embedding of the actual state space into a Euclidean space might be necessary: For instance, to include constant functions, one adds a dimension to the space and amends all state vectors by a constant 1 in this dimension.

It turns out that the sub-gradient envelope operation $\mathcal{S}_G$ acting on convex piecewise linear functions corresponds to a certain row-rearrangement operator Υ_G acting on the matrix representatives of these functions, in the sense that

$$f \sim F \quad \Rightarrow \quad \mathcal{S}_G f \sim \Upsilon_G[F].$$

Such row-rearrangement operator Υ_G, associated with the grid

$$G = \{g^1, \dots, g^m\} \subset \mathbb{R}^d$$

acts on each matrix F with d columns as follows:

$$(\Upsilon_G[F])_{i,\cdot} = F_{\mathrm{argmax}(Fg^i),\cdot} \qquad \text{for all } i = 1, \dots, m. \tag{28}$$

If piecewise linear and convex functions $(f_i)_{i=1}^n$ are given in terms of their matrix representatives $(F_{i,\cdot})_{i=1}^n$, such that

$$f_k \sim F^k, \quad k = 1, \dots, n.$$

then it holds that

$$\mathcal{S}_G(\sum_{k=1}^n f_k) \quad \sim \quad \sum_{k=1}^n \Upsilon_G[F^k] \tag{29}$$

$$\mathcal{S}_G(\bigvee_{k=1}^n f_k) \quad \sim \quad \Upsilon_G[\sqcup_{k=1}^n F^k] \tag{30}$$

$$\mathcal{S}_G(f_k(W\cdot) \quad \sim \quad \Upsilon_G[F^k W] \qquad k = 1, \dots, n, \tag{31}$$

where the operator $\sqcup$ denotes binding matrices by rows (for details, we refer the reader to [22]) and to ([23]). The algorithms presented there use the properties (29)–(31) to calculate approximate value functions in terms of their matrix representatives as follows:

Pre-calculations: Given a grid $G = \{g^1, \dots, g^m\}$, implement the row rearrangement operator Υ_G and the row maximization operator $\sqcup_{a \in A}$. Determine a distribution sampling $(W_t^{(k)})_{k=1}^n$ of each disturbance W_t with corresponding weights $(\nu_t^{(k)})_{k=1}^n$ for $t = 1, \dots, T$. Given reward functions $(r_t)_{t=0}^{T-1}$ and scrap value r_T, assume that the matrix representatives of their sub-gradient envelopes are given by

$$R_t(p, a) \sim \mathcal{S}_G r_t(p, \cdot, a), \qquad R_T(p) \sim \mathcal{S}_G r_T(p, \cdot)$$

for $t = 0, \dots, T-1$, $p \in P$ and $a \in A$. The matrix representatives of each double-modified value function

$$v_t^{(m,n)}(p, \cdot) \sim V_t(p) \quad \text{for } t = 0, \dots, T, \, p \in P$$

are obtained via the following matrix-form of the approximate backward induction (also depicted in the Algorithm 1.

Initialization: Start with the matrices

$$V_T(p) = R_T(p), \qquad \text{for all } p \in P.$$

Recursion: For $t = T-1, \dots, 0$ and for $p \in P$ calculate

$$V_{t+1}^E(p, a) \quad = \quad \sum_{p' \in P} \sum_{k=1}^n \nu_{t+1}^{(k)} \alpha_{p,p'}^a(\cdot) \star \Big[V_{t+1}(p') W_{t+1}^{(k)}\Big], \tag{32}$$

$$V_t(p) \quad = \quad \sqcup_{a \in A} \Big(R_t(p, a) + V_{t+1}^E(p, a)\Big). \tag{33}$$

Algorithm 1: Value Function Approximation.

```
for $p \in P$ do
    $V_T(p) \sim \mathcal{S}_G r_T(p,.)$,    $V_T(p) \leftarrow \Upsilon_G[V_T(p)]$,
    for $a \in A, t = 0, \dots, T$ do
        $R_t(p,a) \sim \mathcal{S}_G r_t(p,.,a)$,    $R_t(p,a) \leftarrow \Upsilon_G[R_t(p,a)]$
    end
end
for $t \in \{T-1, \dots, 0\}$ do
    for $p \in P$ do
        for $a \in A$ do
            $V^E_{t+1}(p,a) \leftarrow \sum_{p' \in P} \sum_{k=1}^{n} \nu^{(k)}_{t+1}(k) \alpha^a_{p,p'}(\cdot) \star \left[V_{t+1}(p') W^{(k)}_{t+1}\right]$
        end
    end
    for $p \in P$ do
        $V_t(p) \leftarrow \sqcup_{a \in A} \big(R_t(p,a) + V^E_{t+1}(p,a)\big)$
    end
end
```

Here, the term $\alpha^a_{p,p'}(\cdot) \star V_{t+1}(p')W^{(k)}_{t+1}$ stands for the matrix representative of the sub-gradient envelope of the product function

$$\mathcal{S}_G\Big[z \mapsto \alpha^a_{p,p'}(z) \cdot \max\Big(V_{t+1}(p')W^{(k)}_{t+1} z\Big)\Big]$$

which must be calculated from both factors using the product rule. The product rule has to be modified due to the assumption that in our context, the sub-gradient $\nabla_g f$ of a function f at a point g is an affine–linear function, which represents the Taylor approximation developed at g to the linear term. For such sub-gradients, the product is given by $\nabla_g(f_1 f_2) = (\nabla_g f_1) f_2(g) + f_1(g)(\nabla_g f_2) - f_1(g) f_2(g)$. The concrete implementation of this operation depends on how the matrix representative of a constant function is expressed. In Section 6.1, we provide a code which realizes such a product rule, based on the assumption that the state space is represented by probability vectors.

Having calculated matrix representatives $(V^E_t)_{t=0}^T$, approximations to expected value functions are obtained as

$$v^E_{t+1}(p,z,a) \quad = \quad \max(V^E_{t+1}(p,a)z) \tag{34}$$

for all $z \in \mathbb{R}^d$, $t = 0, \dots, T-1$, $a \in A$ and $p \in P$. Furthermore, an approximately optimal strategy $(\pi_t)_{t=0}^{T-1}$ is obtained for $t = 0, \dots, T-1$ by

$$\pi_t(p,z) = \mathrm{argmax}_{a \in A}(r_t(p,z,a) + v^E_{t+1}(p,z,a)), \quad p \in P,\ z \in \mathbb{R}^d. \tag{35}$$

In what follows, we apply this technique to our regime-switching Markov decision problems.

5. HMM-Modulated MDP as a Convex Switching Problem

Now, we turn to the main step—an appropriate extension of the state space $P \times \mathcal{S}$ (as introduced in Section 3). This procedure will allow treating our regime-switching Markov decision problems by numerical methodologies described in the previous section.

With notations and conventions of Section 3, we consider the so-called positively homogeneous function extensions: A function $\tilde{v} : P \times \mathbb{R}^d_+ \to \mathbb{R}$ is called positively homogeneous if $\tilde{v}(p, cx) = c\tilde{v}(p,x)$ holds for all $c \in \mathbb{R}_+$ and $(p,x) \in P \times \mathbb{R}^d_+$. Obviously, for each $v : P \times \widehat{S} \to \mathbb{R}$ the definition

$$\tilde{v}(p,z) = \|z\| v(p, \frac{z}{\|z\|}), \qquad (p,z) \in P \times \mathbb{R}^S_+$$

yields a positively-homogeneous extension of v.

Given the stochastic kernel (18), we construct a probability space supporting a sequence $(Y_t)_{t=1}^T$ of independent identically distributed random variables, each following the same distribution μ in order to introduce the random matrices (disturbances)

$$W_t = \Gamma^\top \mathcal{V}(Y_t), \qquad t = 1, \ldots, T.$$

These disturbances are used to define the following stochastic kernels in $P \times \mathbb{R}^S_+$:

$$\tilde{\mathcal{K}}^a_t \tilde{v}(p,z) = \sum_{p' \in P} \alpha^a_{p,p'}\left(\frac{z}{\|z\|}\right)\mathbb{E}(\tilde{v}(p', W_{t+1}z)), \qquad (p,z) \in P \times \mathbb{R}^S_+,\ t = 0, \ldots, T-1 \tag{36}$$

acting on all functions $\tilde{v} : P \times \mathbb{R}^S_+ \to R$ where the above expectations are well-defined. A direct verification shows that for each $a \in A$ and $t = 0, \ldots, T-1$ it holds that

$$\begin{array}{c}\text{if } \tilde{v} \text{ is a positively homogeneous extension of } v \text{ then} \\ \tilde{\mathcal{K}}^a_t \tilde{v} \text{ is a positively homogeneous extension of } \mathcal{K}^a_t v.\end{array} \tag{37}$$

The kernels (36) satisfy the linear dynamics assumption (21) required in (22) and define a control problem of convex switching type whose value functions also solve the underlying regime-switching Markov decision problem.

Proposition 1. *Given a regime modulated Markov decision problem whose dynamics are defined by the stochastic kernel (18) with control costs given by (19) and (20), consider the value functions $(v_t)_{t=0}^T$ returned by the corresponding backward induction*

$$\begin{aligned} v_T(p,\hat{s}) &= r_T(p,\hat{s}), \\ v_t(p,\hat{s}) &= \max_{a \in A}(r_t(p,\hat{s},a) + \mathcal{K}^a_t v_{t+1}(p,\hat{s})), \quad p \in P, \hat{s} \in \hat{S},\ t = T-1, \ldots, 0. \end{aligned} \tag{38}$$

Moreover, consider functions $(\tilde{v}_t)_{t=0}^T$ returned by

$$\begin{aligned} \tilde{v}_T(p,z) &= \tilde{r}_T(p,z), \\ \tilde{v}_t(p,z) &= \max_{a \in A}(\tilde{r}_t(p,z,a) + \tilde{\mathcal{K}}^a_t \tilde{v}_{t+1}(p,z)), \quad p \in P, z \in \mathbb{R}^S_+,\ t = T-1, \ldots, 0. \end{aligned} \tag{39}$$

where $\tilde{r}_t$, $\tilde{r}_T$ are positively homogeneous extensions of r_t, r_T and $\tilde{\mathcal{K}}^a_t$ is from (36). Then for $t = 0, \ldots, T$, it holds that

$$\tilde{v}_t \text{ is positively homogeneous extension of } v_t \tag{40}$$

Proof. Let us prove (40) inductively. Starting at $t = T$, assertion (40) holds since $\tilde{r}_T$ is a positively homogeneous extension of r_T. Having assumed that (40) holds for $t+1$ with a positively homogeneous $\tilde{v}_{t+1}$, we apply observation (37) to conclude that

$$\tilde{\mathcal{K}}^a_t \tilde{v}_{t+1} \text{ is a positively homogeneous extension of } \mathcal{K}^a_t v_{t+1}$$

thus adding $\tilde{r}_t(\cdot,\cdot,a)$, which is a positively homogeneous extension of $r_t(\cdot,\cdot,a)$ and maximizing over $a \in A$ yields (40).

To finish the proof, we verify (37). Given v, with a positively homogeneous extension $\tilde{v}$, for $p \in P$ and $z \in \mathbb{R}^S_+$ it holds that

$$\begin{aligned} \tilde{\mathcal{K}}^a_t \tilde{v}(p,z) &= \sum_{p' \in P} \alpha^a_{p,p'}(\frac{z}{\|z\|})\mathbb{E}(\tilde{v}(p', \Gamma^\top \mathcal{V}(Y_{t+1})z)) \\ &= \sum_{p' \in P} \alpha^a_{p,p'}(\frac{z}{\|z\|}) \int_Y \tilde{v}\Big(p', \Gamma^\top \mathcal{V}(y')z\Big)\|\mu(\mathrm{d}y') \qquad (41) \\ &= \sum_{p' \in P} \alpha^a_{p,p'}(\frac{z}{\|z\|}) \int_Y v\left(p', \frac{\Gamma^\top \mathcal{V}(y')z}{\|\mathcal{V}(y')z\|}\right)\|\mathcal{V}(y')z\|\mu(\mathrm{d}y'). \qquad (42) \end{aligned}$$

From the expression (41) we conclude that $\tilde{\mathcal{K}}^a_t \tilde{v}$ is indeed positively homogeneous. Setting $z = \widehat{s} \in \widehat{S}$, we observe $\tilde{\mathcal{K}}^a_t \tilde{v}(p, \widehat{s}) = \mathcal{K}^a_t v(p, \widehat{s})$, meaning that $\tilde{\mathcal{K}}^a_t \tilde{v}$ is a function extension of $\mathcal{K}^a_t v$. $\square$

6. Algorithm Implementations and Performance Analysis

The stylized algorithm presented in Algorithm 1 is appropriate for problems whose scale is similar to that of the illustration provided in the next section. In this example, although we have used a relatively slow scripting language R, all calculations are performed within a few seconds. This shows a practical relevance of such implementations for small and medium-size applications. However, to address larger problems, a significant increase in calculation performance is needed and can be achieved by approximations, which are based on a standard technique from big data analysis, the so-called *next-neighbor search*. A realization of this concept within a package for the statistical language R is described in [26]. With all critical parts of the algorithm written in C, this implementation shows a reasonable performance which is examined and discussed in [26]. For technical details, we address the reader to [23,27]. Let us merely highlight the main idea here. The point is that the computational performance of our approach suffers from the fact that most of the calculation time is being spent on matrix rearrangements required by the operator Υ_G. Namely, in order to calculate an expression

$$\sum_{k=1}^{n} \nu^{(k)}_{t+1} \Upsilon_G[V_{t+1}(p) \cdot W_{t+1}(k)] \qquad (43)$$

as in (25), the row-rearrangement Υ_G must be performed n times, once for each disturbance matrix multiplication. This task becomes increasingly demanding for larger values of the disturbance sampling sizes n, particularly in high state space dimensions. Let us omit $t+1$ and p in (43) to clarify the idea of efficiency improvement developed in [23]. This approach focuses on the two major computational problems

$$\begin{array}{c}\text{the rearrangement } \Upsilon_G[VW(k)] \text{ of} \\ \text{large matrices } V \cdot W(k)\end{array} \qquad (44)$$

and

$$\begin{array}{c}\text{the summation of matrices } \Upsilon[V \cdot W(k)] \text{ over} \\ \text{a large index range } k = 1, \ldots, n.\end{array} \qquad (45)$$

It turns out that one can approximate the procedure in (44) by replacing the row-rearrangement operation with an appropriate matrix multiplication using next-neighbor techniques. To address (45) each random disturbance matrix W_t is represented as the linear combination

$$W_t = \bar{W} + \sum_{j=1}^{J} \epsilon_j(t) E(j) \qquad (46)$$

with non-random matrices $\bar{W}$ and $(E(j))_{j=1}^{J}$, and random coefficients $(\varepsilon_j(t))_{j=1}^{J}$ whose dimension J is preferably significantly lower than the dimension of the state space. Both techniques are applicable and save a significant amount of calculations. Only the disturbances $(W_t)_{t=1}^{T}$ are identically distributed, so that all pre-calculations have to be done only once. However, since a detailed discussion of this approximation and its performance gain are out of the scope here, we refer the reader to [23,26,27].

Note that an application of convex switching techniques under partial observations requires some adaptations, due to the non-observable nature of the state space. More precisely, the user must realize a recursive filter to extract believe states from the current information flow, before optimal decisions can be made. In other words, filtering must be followed by the application of the decision policy, returned from the optimization. A stylized realization of this approach is depicted in Figure 1.

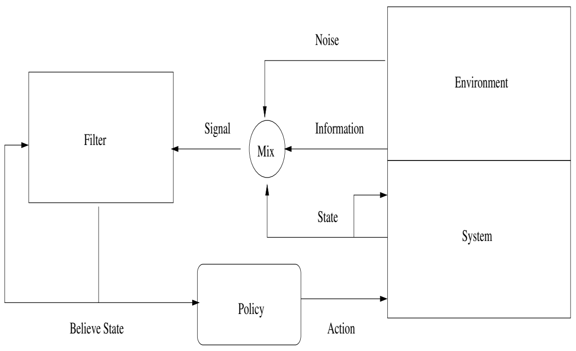

Figure 1. Stylized implementation of the control flow within a realistic application.

6.1. An illustration

Let us consider a typical application of Markov decision theory to agricultural management [28] which addresses a stochastic forest growth model in the context of timber harvesting optimization. The work [28] re-considers the classical results of Faustmann in the framework of random growth and potential ecological hazards (for a derivation of Faustmann's result from 1849 and its discussion, we refer the reader to [28]). The idea is that as a reasonable approximation, one supposes that the only stochastic element is the growth of trees: That is, the timber volume per hectare defines the state. To facilitate the analysis, this timber volume (cubic meter per hectare) is discretized as shown in the first row of the Table 2. Furthermore, the costs of action (in USD per hectare) are shown in the second row, assuming that the action means timber harvest for all states (state 2–state 6) followed by re-forestation with the exception of the bare land (state 1), where there is re-forestation only. There are no costs if no action is taken, as seen from the third row of Table 2.

Table 2. Discrete states and their control costs.

	State 1	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**
volume	0	29	274	530	728	868
acting	–494	–117	3068	6396	8970	10,790
being idle	0	0	0	0	0	0

Given the state space $P = \{1, \ldots, 6\}$ and control costs as defined in Table 2, define the action space $A = \{1, 2\}$, where 1 means being idle and 2 means acting. Assuming decision intervals of 20 years, a Markov decision problem is determined with the action-dependent stochastic matrices

$$\alpha^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.7 & 0.1 & 0 & 0 \\ 0.1 & 0 & 0.1 & 0.7 & 0.1 & 0 \\ 0.1 & 0 & 0 & 0.1 & 0.7 & 0.1 \\ 0.1 & 0 & 0 & 0 & 0.1 & 0.8 \\ 0.1 & 0 & 0 & 0 & 0 & 0.9 \end{bmatrix}, \qquad \alpha^2 = \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the transitions to state 1 (bare land) from higher state in α^1 may be interpreted as a result of a natural disaster, whereas those to a neighboring state may describe a randomness in the forest growth. The matrix α^2 describes the timber harvest followed by re-forestation. The work [28] discusses an optimal policy for this model assuming an infinite time horizon with a discounting which is inferred from the interest rate effects. The optimal policy is compared to that from a non-random growth model, represented by the same parameters with the exception of the free-growth transition α^1, being replaced by

$$\alpha^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This non-random model resembles the well-known results of Faustmann, claiming that the only optimal strategy is a roll-over of harvesting and re-planting after a number of years. In the above Markov decision problem, the discount factor is set to $\kappa = (1+g)^{20} = 0.61$ (corresponding to $g = 2.5\%$ annual interest rates). Here, it turns out that it is optimal to reforest in state 1, do nothing in state 2 and 3, and cut and reforest in states 4, 5, and 6. That is, the resulting roll-over is three times the decision period, 60 years. In contrast, the random model suggests that in the presence of ecological risks, it is optimal to reforest in state 1, do nothing in state 2, and cut and reforest in states 3,4, 5, and 6, giving a shorter roll-over of two decision periods, 40 years.

Using our techniques, the optimal forest management can be refined significantly. Major improvement can be achieved by incorporating all adverse ecological situations into a selection of transition matrices representing a random forest evolution within appropriate regimes. The regime switch can be monitored and estimated using stochastic filtering techniques as described above. For instance, the potential climate change with an anticipated increase of average temperature and extended drought periods can be managed adaptively. In what follows, we consider a simplified numerical example based on the stochastic forest growth model presented above.

First, let us replicate the results of [28] in the following code

```r
rm(list=ls(all=TRUE)) # remove all objects
##########################################
reward<-function(t, a)# define reward function
{  if (a==2) result<-c(-494,  -117,  3068,  6396,  8970,  10790)
else  result<-c(0,  0,  0, 0,  0,  0)
return(result)
}
#########################################
discount<-0.61# introduce discount factor
##############################################
alpha<-array(data=0, dim=c(6,6,2))# set transitions
alpha[,,2]<-matrix(c(0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0,0, 0, 0,
0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0, 0   ), byrow=TRUE, ncol=6)
```

```
alpha[,,1]<-matrix(c(1, 0, 0, 0, 0, 0,
0.1, 0.1, 0.7, 0.1, 0,0,
0.1, 0, 0.1, 0.7, 0.1,0,
0.1, 0, 0,0.1, 0.7, 0.1,
0.1, 0, 0, 0, 0.1, 0.8,
0.1, 0, 0, 0, 0, 0.9  ), byrow=TRUE, ncol=6)
###############################################
bellman<-function(t,val)# define backward induction step
{ container<-array(data=0, dim=dim(alpha)[2:3] )
for (a in 1:dim(alpha)[3]) container[,a]<-reward(t, a) + discount*alpha[,,a]%*%val
result<-cbind(apply(FUN=max, X=container, MARGIN=1),
apply(FUN=which.max, X=container, MARGIN=1) )
}
##############################################
# RUN
#############################################
t_final<-10; val<-reward(t_final, 2)
t<-t_final # initialization
#######################################
while (t>0){# recursion
t<-t-1
result<-bellman(t, val)
val<-result[,1] }
####################################
result # show value function and policy
#[,1] [,2]
#[1,]  1024.141    2
#[2,]  2660.412    1
#[3,]  4586.141    2
#[4,]  7914.141    2
#[5,] 10488.141    2
#[6,] 12308.141    2
```

Indeed, the above plot shows that in the stochastic forest growth model, it is optimal to do nothing in state 2, but harvest and replant in all other states—an optimal roll-over of two periods, 40 years. Running the same algorithm for the deterministic growth, returns the value functions and policies given below—an optimal roll-over over three periods, 60 years.

```
#[1,]  1227.540    2
#[2,]  3012.010    1
#[3,]  4948.456    1
#[4,]  8117.540    2
#[5,] 10691.540    2
#[6,] 12511.540    2
```

Now let us illustrate an adaptive forest management in presence of regime-changing risk. First, let us specify the hidden Markovian dynamics of the information process $(Y_t)_{t=1}^T$. For this, a stochastic matrix $\Gamma = (\Gamma_{s,s'})_{s,s' \in S}$ and a family of measures $(\mu_s)_{s \in S}$ must be specified, according to (15). For simplicity, we suppose that the state comprises two regimes $S = \{s_1, s_2\}$ and that an indirect information $(Y_t)_{t=1}^T$ is observed within the interval $[0, 1]$, after appropriate transformation. For instance, Y_t could measure a percentage of rainy days over a pre-specified sliding window over a past period. Alternatively, Y_t may stand for a specific quantity (average temperature) with distribution function typical for this quantity in the normal regime, applied on its recording. With this assumption, we consider s_1 as a regular regime, under which the observations follow a beta distribution $\beta(s_1)$ with specific parameters defined for the state s_1, whereas in the presence of environmental hazards, in the regime s_2, the observation follows a beta distribution $\beta(s_2)$ with other parameters typical for s_2. Finally, let us suppose that the regime change matrix

$$\Gamma = \begin{bmatrix} \gamma_1, & 1-\gamma_1 \\ 1-\gamma_2 & \gamma_2 \end{bmatrix}$$

describes a situation where the regular regime is more stable $1 \geq \gamma_1 > \gamma_2 > 0$. The following code illustrates a simulation of such observations and the corresponding filtered information with results of filtering procedure depicted in Figure 2

```r
rm(list=ls(all=TRUE))
set.seed(69)
gamma1<-0.99
gamma2<-0.9
Gamma<-matrix(data=c(gamma1, 1-gamma1, 1-gamma2, gamma2), nrow=2, byrow=TRUE)
shapes<-matrix(data=c(1, 3, 3, 1), nrow=2)
y<-function(regime)
{  return(rbeta(n=1,shapes[regime,1],shapes[regime,2]))}
dens<-function(observaton)
{ return(c(dbeta(observaton, shapes[1,1],shapes[1,2]),
dbeta(observaton, shapes[2,1],shapes[2,2])))
}
########################################
regimes<-vector(mode="numeric",length=200)
observations<-vector(mode="numeric",length=length(regimes))
regime<-1
for (j in 1:length(regimes)) {
regimes[j]<- which(rmultinom(n=1, size=1, prob = Gamma[regime,])==1)
observations[j]<-y(regime)
regime<-regimes[j]}

plot(regimes-1, type="l", ylim=c(0,1))
points(observations, type="l", col="red")
###########################################
believ<-c(0.5,0.5)
belives<-matrix(data=0,nrow=length(regimes), ncol=ncol(Gamma))
for (j in 1:length(regimes)) {
believ<-t(Gamma)%*%(dens(observations[j])*believ)
believ<-believ/sum(believ)
belives[j,]<-believ
}
points(belives[,2], type="l", col="blue")
```

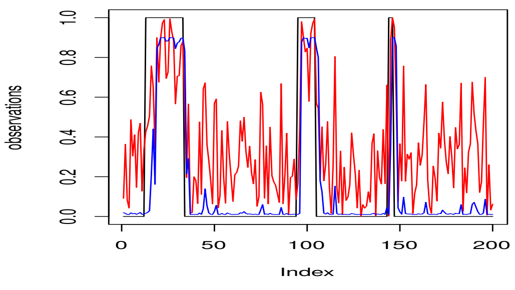

Figure 2. Believe state evolution: Observations depicted by red line, the probability that the system is in the regime 2 by black (true state) and blue (filtered believe) line.

Let us address an implementation of the Algorithm 1. Having defined the appropriate operators

```r
###################################################
rm(list=ls(all=TRUE))
##################################################
##################################################
make_max<-function(L1,L2) # maximization
{ value1<-apply( L1*G, FUN=sum,MARGIN=1)
value2<-apply( L2*G, FUN=sum,MARGIN=1)
L<-L1
L[value1<value2,]<-L2[value1<value2, ]
return(L)}
#################################################
make_max.where<-function(L1,L2) # maximization
{ value1<-apply( L1*G, FUN=sum,MARGIN=1)
value2<-apply( L2*G, FUN=sum,MARGIN=1)
result<-rep(1, length(value1))
result[value1<value2]<-2
return(result)}
#################################################
Y<-function(L) # row-re-arrangement
{return(L[apply(L%*%t(G), FUN=which.max, MARGIN=2),])}
###################################################
make_const<-function(L)
{
```

```
value<-apply( L*G, FUN=sum,MARGIN=1)
matrix(data=value, byrow=FALSE, nrow=grid_size, ncol=regime_size)
}
```

we introduce arrays as data containers

```
####################################################
state_size<-6 # six states of the forest
regime_size<-2 # two regimes
action_size<-2 # two actions
grid_size<-100 # number of grid points
sample_size<-200 # size of distribution sampling
##################################################
G<-array(data=0, dim=c(grid_size, regime_size)) # grid matrix
Gamma<-array(data=0, dim=c(regime_size, regime_size))
alpha<-array(data=0, dim=c(state_size, state_size, regime_size, action_size))
Alpha<-array(data=0, dim=c( grid_size, regime_size, state_size, state_size, action_size))
Alphaconst<-array(data=0, dim=c( grid_size, regime_size, state_size, state_size, action_size))
scrap<-array(data=0, dim=c(1, state_size)) # scrap matrix
reward<-array(data=0, dim=c(1, state_size, action_size)) # reward matrix
distrsample<-array(data=0, dim=c(1,   sample_size)) # distribution sampling
disturbance <-array(data=0, dim=c(regime_size, regime_size,  sample_size)) # disturbances
Vfun<-array(data=0, dim=c(grid_size, regime_size, state_size))
Efun<-array(data=0, dim=c(grid_size, regime_size, state_size))
Cfun<-array(data=0, dim=c(grid_size, regime_size, state_size, action_size))
Reward<-array(data=0, dim=c(grid_size, regime_size, state_size, action_size))
Scrap<--array(data=0, dim=c(grid_size, regime_size, state_size))
Decision<-array(data=0, dim=c(grid_size, state_size))
```

these containers are filled with appropriate quantities:

```
G[,1]<-seq(from=0, to=1, length=grid_size)
G[,2]<-1-G[,1]  # fill the grid with equidistant points
gamma1<-0.99 # regime 1 is stable, regular
gamma2<-0.99 # regime 2 is unstable, hazard
Gamma<-matrix(data=c(gamma1, 1-gamma1, 1-gamma2, gamma2), nrow=2, byrow=TRUE)
alpha[,,1,2]<-matrix(c(0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0,0, 0, 0,
0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0, 0   ), byrow=TRUE, ncol=6)
alpha[,,1,1]<-matrix(c(1, 0, 0, 0, 0, 0,
0.1, 0.1, 0.7, 0.1, 0,0,
0.1, 0, 0.1, 0.7, 0.1,0,
0.1, 0, 0,0.1, 0.7, 0.1,
0.1, 0, 0, 0, 0.1, 0.8,
0.1, 0, 0, 0, 0, 0.9  ), byrow=TRUE, ncol=6)
alpha[,,2,2]<-matrix(c(0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0, 0, 0,0,
0.1, 0.9, 0,0, 0, 0,
0.1, 0.9, 0, 0, 0, 0,
0.1, 0.9, 0, 0, 0, 0   ), byrow=TRUE, ncol=6)
alpha[,,2,1]<-matrix(c(1, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0,0,
0, 0, 0, 1, 0,0,
0, 0, 0,0, 1, 0,
0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 1  ), byrow=TRUE, ncol=6)
for (a in 1:action_size)
for (i in 1:grid_size)
for (j in 1:regime_size)
Alpha[i,j,,,a]<-alpha[,,j,a]
for (a in 1:action_size)
for (p1 in 1:state_size)
for (p2 in 1:state_size)
Alphaconst[,,p1,p2,a]<-make_const(Alpha[,,p1,p2,a])
scrap[1,]<-c(-494,  -117,  3068,  6396,  8970,  10790)
reward[1,,2]<-c(-494,  -117,  3068,  6396,  8970,  10790)
for (p in 1:state_size)
for (a in 1:action_size)
Reward[,,p,a]<-Y(matrix(reward[1,p,a], nrow=1, ncol=regime_size)) # convex lin comb like in Alpha
for (p in 1:state_size)
Scrap[,,p]<-Y(matrix(scrap[1,p], nrow=1, ncol=regime_size)) # convex lin comb like in Alpha
distrsample[1,]<-seq(from=0, to=1, length=sample_size)
shapes<-matrix(data=c(1, 3, 3, 1), nrow=2)
dens<-function(observaton)
```

```
{ return(c(dbeta(observaton, shapes[1,1],shapes[1,2]),
dbeta(observaton, shapes[2,1],shapes[2,2])))
}
kernel<-array(data=0, dim=c(regime_size, sample_size))
for (y in 1:sample_size)
kernel[,y]<-dens(distrsample[y])
for (j in 1:regime_size)
kernel[j,]<-kernel[j, ]/sum(kernel[j,])
for (y in 1:sample_size)
disturbance[,,y]<- t(Gamma)%*%diag(kernel[,y])
discount<-0.61# introduce discount factor
```

After such initialization, the backward induction is run:

```
Vfun[,,]<-Scrap[,,] ; tt<-10 # value function and time  horizon

while (tt>0){
tt<-tt-1
for (p in 1:state_size) {
Efun[,,p]<-0
for (j in 1:sample_size)   Efun[,,p]<-Y(Efun[,,p])+Y(Vfun[,,p]%*%disturbance[,,j])}

for (a in 1:action_size)
for (p1 in 1:state_size) {Cfun[,,p1,a]<-0
for (p2 in 1:state_size) {Econst<-make_const(Efun[,,p2])
Cfun[,,p1,a]<- Cfun[,,p1,a]+Alphaconst[,,p1,p2,a]*Efun[,,p2] +
Alpha[,,p1,p2,a]*Econst -Alphaconst[,,p1,p2,a]*Econst}}

for (p in 1:state_size)
for (a in 1:action_size)
Cfun[,,p,a]<-Y(Reward[,,p,a]) + discount*Y(Cfun[,,p,a])

for (p in 1:state_size){
Vfun[,,p]<-make_max( Cfun[,,p,1],Cfun[,,p,2]) # maximize
Decision[,p]<-make_max.where( Cfun[,,p,1],Cfun[,,p,2])}
print(tt)
}

plot(x=G[,2], y=apply(X=Vfun[,,2]*G, FUN=sum, MARGIN=1), col='black', type='l',  xlab=NA, ylab=NA)
plot(x=G[,2], y=Decision[,3], col='black', type='l',  xlab=NA, ylab=NA)
```

The Figure 3 depicts the value function for the state $p = 2$ depending on the believe state, such that the x- axes is interpreted as the conditioned probability that the system is in the deterministic growth regime. That is, the end points of the graph represent values of the expected return of the optimal strategy under the condition that it is known with certainty that the system starts in random growth (left end) or in deterministic growth (right end). Observe that those values are close to 2660.412 and 3012.010 obtained in the classical Markov decision setting. Naturally, the uncertainty about the current regime yields intermediate values, connected by a convex line.

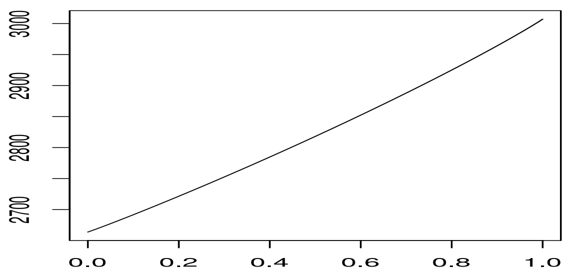

Figure 3. Value function for the state 2 depending on believe probability.

The Figure 4 illustrates the choice of the optimal action for the state $p = 3$ depending on the believe state depicted in the same way as in Figure 3. Again the end points of the graph represent optimal actions conditioned on certainty about the current regime. As expected, the optimal decision switches from $a = 2$ (act) to $a = 1$ (be idle) at some believe probability of approximately 0.55.

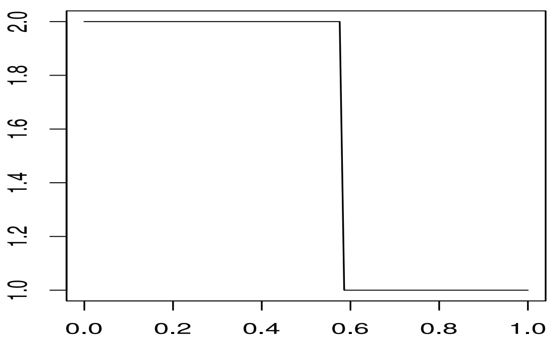

Figure 4. Optimal action for the state 3 depending on believe probability.

7. Conclusions

Having utilized a number of specific features of our problem class, we suggest a simple, reliable, and easy-to-implement algorithm that can provide a basis for rational sequential decision-making under uncertainty. Our results can be useful if there are no historical data about what could go wrong, or when high requirements on risk assessment are proposed. In such situations, we suggest encoding all relevant worst-case scenarios as potential regime changes, making conservative a priori assumptions on regime change probabilities. Using our algorithm, all optimal strategies can be efficiently examined in such a context, giving useful insights about parameter sensitivity and model risk associated with this sensitivity. The author believes that the suggested algorithm can help gaining a better understanding of risks and opportunities in such context.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; Wiley: New York, NY, USA, 1994.
2. Smallwood, R.D.; Sondik, E.J. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Oper. Res.* **1973**, *21*, 1071–1088. [CrossRef]
3. Hinz, J.; Tarnopolskaya, T.; Yee, T. Efficient algorithms of pathwise dynamic programming for decision optimization in mining operations. *Ann. Oper. Res.* **2020**, *286*, 583–615. [CrossRef]
4. Tsiropoulou, E.E.; Kastrinogiannis, T.; Symeon, P. Uplink Power Control in QoS-aware Multi-Service CDMA Wireless Networks. *J. Commun.* **2009**, *4*. [CrossRef]
5. Huang, X.; Hu, F.; Ma, X.; Krikidis, I.; Vukobratovic, D. Machine Learning for Communication Performance Enhancement. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 3018105:1–3018105:2. [CrossRef]
6. Bäuerle, N.; Rieder, U. *Markov Decision Processes with Applications to Finance*; Springer: Heidelberg, Germany, 2011. [CrossRef]
7. Powell, W.B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*; Wiley: Hoboken, NJ, USA, 2007.
8. Belomestny, N.; Kolodko, A.; Schoenmakers, J. Regression methods for stochastic control problems and their convergence analysis. *SIAM J. Control Optim.* **2010**, *48*, 3562–3588. [CrossRef]
9. Carriere, J.F. Valuation of the Early-Exercise Price for Options Using Simulations and Nonparametric Regression. *Insur. Math. Econ.* **1996**, *19*, 19–30. [CrossRef]
10. Egloff, D.; Kohler, M.; Todorovic, N. A dynamic look-ahead Monte Carlo algorithm. *Ann. Appl. Probab.* **2007**, *17*, 1138–1171. [CrossRef]
11. Tsitsiklis, J.N.; Van Roy, B. Regression Methods for Pricing Complex American-Style Options. *IEEE Trans. Neural Netw.* **2001**, *12*, 694–703. [CrossRef]
12. Tsitsiklis, J.; Roy, B.V. Optimal Stopping of Markov Processes: Hilbert Space, Theory, Approximation Algorithms, and an Application to Pricing High-Dimensional Financial Derivatives. *IEEE Trans. Automat. Contr.* **1999**, *44*, 1840–1851. [CrossRef]
13. Longstaff, F.; Schwartz, E. Valuing American options by simulation: A simple least-squares approach. *Rev. Financ. Stud.* **2001**, *14*, 113–147. [CrossRef]

14. Ormoneit, D.; Glynn, P. Kernel-Based Reinforcement Learning. *Mach. Learn.* **2002**, *49*, 161–178. [CrossRef]
15. Ormoneit, D.; Glynn, P. Kernel-Based Reinforcement Learning in Average-Cost Problems. *IEEE Trans. Automat. Contr.* **2002**, *47*, 1624–1636. [CrossRef]
16. Fan, J.; Gijbels, I. *Local Polynomial Modelling and Its Applications*; Chapman and Hall: London, UK, 1996.
17. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-Dynamic Programming*; Athena Scientific: Nashua, NH, USA, 1996.
18. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artific. Intell.* **1998**, *101*, 99–134. [CrossRef]
19. Lovejoy, W.S. Computationally Feasible Bounds for Partially Observed Markov Decision Processes. *Operat. Res.* **1991**, *39*, 162–175. [CrossRef]
20. Shani, G.; Pineau, J.; Kaplow, R. A survey of point-based POMDP solvers. *Autonom. Agents Multi-Agent Syst.* **2013**, *27*, 1–51. [CrossRef]
21. Pineau, J.; Gordon, G.; Thrun, S. Point-based value iteration: An anytime algorithm for POMDPs. In Proceedings of the 18th International IJCAI'03: Joint Conference on Artificial Intelligence, Acapulco, Mexico, 9–15 August 2003; pp. 1025–1032.
22. Hinz, J. Optimal stochastic switching under convexity assumptions. *SIAM J. Contr. Optim.* **2014**, *52*, 164–188. [CrossRef]
23. Hinz, J.; Yap, N. Algorithms for optimal control of stochastic switching systems. *Theor. Prob. Appl.* **2015**, *60*, 770–800. [CrossRef]
24. Hinz, J.; Yee, J. Optimal forward trading and battery control under renewable electricity generation. *J. Bank. Financ.* **2017**. [CrossRef]
25. Hinz, J.; Yee, J. Stochastic switching for partially observable dynamics and optimal asset allocation. *Int. J. Contr.* **2017**, *90*, 553–565. [CrossRef]
26. Hinz, J.; Yee, J. Rcss: R package for optimal convex stochastic switching. *R J.* **2018**. [CrossRef]
27. Hinz, J.; Yee, J. Algorithmic Solutions for Optimal Switching Problems. In Proceedings of the 2016 Second International Symposium on Stochastic Models in Reliability Engineering, Life Science and Operations Management (SMRLO), Beer Sheva, Israel, 15–18 February 2016; pp. 586–590. [CrossRef]
28. Buongiorno, J. Generalization of Faustmann's Formula for Stochastic Forest Growth and Prices with Markov Decision Process Models. *For. Sci.* **2001**, *47*, 466–474. [CrossRef]