Methods in Ecology and Evolution

British Ecological Society

*Methods in Ecology and Evolution* 2013, **4,** 872–884 doi: 10.1111/2041-210X.12082

REVIEW

# Complex decisions made simple: a primer on stochastic dynamic programming

**Lucile Marescot[1]*, Guillaume Chapron[2], Iadine Chadès[3], Paul L. Fackler[4], Christophe Duchamp[5], Eric Marboutin[6] and Olivier Gimenez[1]**

[1]*Centre d'Ecologie Fonctionnelle et Evolutive, UMR 5175, campus CNRS, 1919 Route de Mende, 34293 Montpellier, France;* [2]*Grimsö Wildlife Research Station, Department of Ecology, Swedish University of Agricultural Sciences, 73091 Riddarhyttan, Sweden;* [3]*CSIRO Ecosystem Sciences, 41 Boggo Road Dutton Park QLD 4102, Australia;* [4]*Department of Agricultural and Resource Economics, North Carolina State University, Raleigh, NC 27695-8109, USA;* [5]*Office National de la Chasse et de la Faune Sauvage, CNERA prédateurs et animaux déprédateurs, Parc Micropolis, 05000 Gap, France; and* [6]*Office National de la Chasse et de la Faune Sauvage, ZI Mayencin, 5 allée de Béthléem, 38610 Gières, France*

## Summary

**1.** Under increasing environmental and financial constraints, ecologists are faced with making decisions about dynamic and uncertain biological systems. To do so, stochastic dynamic programming (SDP) is the most relevant tool for determining an optimal sequence of decisions over time.

**2.** Despite an increasing number of applications in ecology, SDP still suffers from a lack of widespread understanding. The required mathematical and programming knowledge as well as the absence of introductory material provide plausible explanations for this.

**3.** Here, we fill this gap by explaining the main concepts of SDP and providing useful guidelines to implement this technique, including R code.

**4.** We illustrate each step of SDP required to derive an optimal strategy using a wildlife management problem of the French wolf population.

**5.** Stochastic dynamic programming is a powerful technique to make decisions in presence of uncertainty about biological stochastic systems changing through time. We hope this review will provide an entry point into the technical literature about SDP and will improve its application in ecology.

**Key-words:** *Canis lupus*, decision-making techniques, markov decision process, optimization methods, stochastic dynamic programming

## Introduction

Numerous problems in ecology involve making decisions about the best option among a set of competing strategies. These so-called optimization problems can be solved using mathematical procedures such as linear programming (Nash & Sofer 1996) which allows the determination of maximum benefits or minimum costs given some objectives and under some constraints for deterministic systems assumed at equilibrium. If uncertainty in the dynamic of the system needs to be accounted for, a Markov decision process (MDP, Puterman 1994; Williams 2009) model is usually adopted. 'MDPs are models for sequential decision making when outcomes are uncertain' (Puterman 1994). MDPs are made of two components: Markov chains that model the uncertain future states of the system given an initial state and a decision model. First, a MDP is a Markov chain in which the system undergoes successive transitions from one state to another through time. For example, these state transitions can correspond to the change of a population size from 1 year to the next. In Markov chains, the transitions to future states only depend on the current state of the system. In other words, the state of the system at time step $t$ provides sufficient information to predict the states of the system at time step $t$+1. Second, a MDP involves a decision-making process in which an action is being implemented at each sequential state transition. In the conservation and wildlife management literature, the phrase stochastic dynamic programming (SDP) is often used to refer to both the mathematical model (MDP) and its solution techniques (SDP *per se*). MDPs are usually modelled and solved by going through several successive steps: defining the different objectives and formalizing them as a mathematical function of costs and/or benefits (Williams, Nichols & Conroy 2002); defining possible states of the system, monitoring the system and making statistical inference on system behaviour (Nichols & Williams 2006); defining a set of alternative actions that influence the performance of the system; building a dynamic model to describe the system transitions from one state to another after implementing every possible decision; and finally determining the optimal

*Corresponding author. E-mail: olivier.gimenez@cefe.cnrs.fr

strategy that is the set of decisions that is expected to best fulfil the objectives over time (Runge 2011). These objectives are formalized in a utility function that prioritizes some desired outcomes by evaluating the benefits of any decision for the system (Williams, Nichols & Conroy 2002). MDP models highlight the trade-off between obtaining current utility and altering the opportunities to obtain utility in the future. Such problems abound in ecology because decisions taken today often have important implications for the future behaviour of biological systems.

Stochastic dynamic programming is an optimization technique used to solve MDPs and is appropriate for the nonlinear and random processes involved in many biological systems. While the time dimension is often neglected in optimization procedures such as classical linear or nonlinear programming, SDP determines state-dependent optimal decisions that vary over time (Williams, Nichols & Conroy 2002). Finally, SDP is acknowledged to be one of the best tools for making recurrent decisions when coping with uncertainty inherent to biological systems (Possingham 1997, 2001; Wilson *et al.* 2006; Chadès *et al.* 2011).

The principle of SDP relies on the partitioning of a complex problem in simpler subproblems across several steps that, once solved, are combined to give an overall solution (Mangel & Clark 1988; Lubow 1995; Clark & Mangel 2000). SDP was first developed and used in applied mathematics, economics and engineering (Bellman 1957; Intriligator 1971) and has gained attention in ecology (Mangel & Clark 1988; Shea & Possingham 2000). A pioneer use of SDP was in behavioural ecology to determine individuals' breeding and foraging strategies maximizing fitness (Houston *et al.* 1988; Mangel & Clark 1988; Ludwig & Rowe 1990). Early work in resource management included applications to pest control (Winkler 1975) and fisheries management (Walters 1975; Reed 1979). In conservation biology, SDP has been successfully used to produce evidence-based management recommendations (optimization of resources allocation: Westphal *et al.* 2003; Martin *et al.* 2007; Chadès *et al.* 2011; management of natural resources in the context of global change: Martin *et al.* 2011). In forestry, SDP allowed achieving a balance between the protection of biological diversity and sustainable timber production (Lembersky & Johnson 1975; Teeter, Somers & Sullivan 1993; Richards, Possingham & Tizard 1999). Stochastic dynamic programming has also been implemented in various studies aiming at controlling the spread of weeds, pests or diseases (Shea, Thrall & Burdon 2000; Baxter & Possingham 2011; Pichancourt *et al.* 2012), to determine the best water management policies (Martin *et al.* 2009) or to enhance the efficiency of a biocontrol agent (Shea & Possingham 2000). In wildlife management, SDP has often been used to find the optimal rates for harvesting populations (Johnson *et al.* 1997; Milner-Gulland 1997; Spencer 1997; Martin *et al.* 2010).

Despite the flexible nature of SDP and its ability to solve important decision-making problems in ecology, its transfer to ecologists is difficult. One reason for the slow uptake is the mathematical knowledge required for SDP to be implemented. Here, we provide a primer on SDP for ecologists. We introduce the main concepts of SDP, provide a step-by-step procedure to implement dynamic programming in a deterministic system and illustrate how to make decisions in the presence of uncertainty. We demonstrate the applicability of SDP by applying this approach to data from a wolf population controlled by culling. We provide R code to run the models as well as procedures in specialized toolboxes implementing SDP that can conveniently be amended for one's own purposes.

## The six steps of stochastic dynamic programming

The aim of SDP is to find the solution of an optimization problem based on the 'principle of optimality' which states that 'an optimal policy has the property that, whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regards to the state resulting from the first decision' (Bellman 1957). The principle of optimality allows us to consider a static problem for the current period by assuming that all future decisions will be made optimally. The effect of the current action thus contributes to both current utility and to future utility through its effect on the future state of the system. In this way, SDP finds a strategy that balances current rewards with future opportunities. Stochastic dynamic programming is the technique used to solve a Markov decision problem. One can conceive solving a Markov decision problem through six steps described below. Notations are gathered in Table 1, and a non-exhaustive list of studies that have used SDP is given in Table 2.

The first step defines the optimization objective of the problem. An objective must be specific to the problem, acceptable by involved actors, achievable, defined over a period of time also called time horizon, and measurable with a function that is related to the system states and actions. This function, called utility, gives the reward for the outcome of any action applied to a certain state (Williams, Nichols & Conroy 2002). Several objectives can be defined depending on the type of ecological problem we are investigating, but an optimization objective must be defined as maximization or minimization of a function over a time horizon (Puterman 1994; Converse *et al.* 2012). The time horizon can be defined as finite or infinite. For many

**Table 1.** Notation used in dynamic programming.

| Variable | Notation | Nature |
|---|---|---|
| State variable | $X_t$ | Vector indexed by time |
| Control action | $A_t$ | Vector indexed by time |
| Time | $t$ | Index |
| Optimal action | $\pi^*$ | Vector of length the number of states at time $t$ |
| Utility | $U(X_t, A_t)$ | Function of the states and actions |
| Transition probability | $P(X_{t+1}\|X_t, A_t)$ | Matrix (number of states at $t$, number of states at $t$+1) |
| Value | $V(X_t)$ | Vector of length the number of states at $t$ |
| Discount factor | $\beta$ | Real number between 0 and 1 |

**Table 2.** Non-exhaustive list of studies using stochastic dynamic programming.

| Study | State variable | Objective | Actions | Utility function |
|---|---|---|---|---|
| Shea & Possingham (2000) | Site level of colonization: empty, insecure, established | Biocontrol agent colonizing as many sites and as quickly as possible | Many agents released in small patches Few agents in several patches Mix of both strategies | Number of established sites |
| Venner *et al.* (2006) | Energy supply of the orb-weaving spider | Optimize fitness by maximizing the energy brought by breeding and foraging while minimizing predation and starvation risks | Web-building choice possible web size. | Balance between energy gained from eggs laid and prey caught on the web and energy cost from starvation and from predation risks. |
| Runge & Johnson (2002) | Pre-breeding abundance of ducks | Find the optimal harvest rate given several recruitment and survival functions | Harvest rate | Total number of harvest accumulated through time |
| Martin *et al.* (2010) | Female raccoon abundance Oyster productivity | Maintain Oystercatcher productivity above a level necessary for population recovery while minimizing raccoon removal. | Harvest rate in each age class | Total number of raccoons after harvest with a penalty factor when oyster productivity goes below a threshold |
| Milner-Gulland (1997) | Saiga antilope abundance Proportion of males and females | Maximize monetary yield while preserving the saiga population already threatened by drought | Harvest rate Proportion of males in the harvest | Annual monetary yield from game hunting, given the price of the meat, the horn and management costs |

| Study | Dynamic model | Optimization | Last value | Uncertainty |
|---|---|---|---|---|
| Shea & Possingham (2000) | Colonization, extinction, establishment in insecure sites | Backward iteration T = 10 | Unknown | Probability of establishment and of local extinction |
| Venner *et al.* (2006) | Discrete Markov model describing the transition energy state of a spider from t to $t + 1$ given the choice of web-building of individuals. | Value iteration over an infinite time horizon | Lifetime fitness given its energy state time horizon is expected to be 1 | Probability to catch a prey and predation risks |
| Runge & Johnson (2002) | Reproduction Harvest Natural mortality | Value iteration Infinite time horizon (convergence criterion was no change of state-dependent policy for more than 4 years) No discount rate | No values were assigned to the terminal state of the process $V(X_T)$=0 | Structural uncertainty Recruitment functions (linear, exponential, hyperbolic) Survival functions (constant, logistic, compensatory) |
| Martin *et al.* (2010) | Model structured in 3 age classes (raccoon population) Log-linear relationship between oyster productivity and total number of raccoons. | Backward iteration iterating at most 100 time steps until a stable policy was maintained for 15 time periods | The expected abundance range of raccoon | Environmental stochasticity Parameter uncertainty |
| Milner-Gulland (1997) | Model structure in age and sex classes with density-dependent effects on survival | Value iteration infinite horizon | Expected future yield at time horizon is 1 | Environmental stochasticity and partial controllability |

resource problems, choosing the time horizon is quite challenging and depends on a number of factors. First, there may be mandated constraints on a problem. Conservation and management programmes are often planned on a limited time and budget, and are bounded by political decisions also taken at regular time intervals. For instance, the conservation status of species listed under Appendix S2 of the Habitat Directive is evaluated every 5 years by the European Commission (92/43/EEC). As a consequence, some governments evaluate every 5 years decisions related to management of wildlife and habitats present within their territory (Meedat – Map 2008). For private decisions, a finite horizon is often appropriate for situations in which firms hold time-limited rights to extract resources. Finite horizons should be used carefully in situations where they are arbitrary specified. It is very possible that the 'optimal' decision as the time horizon approaches will reflect

only very short run goals. For example, a conservation problem that penalizes failure to meet a target performance level at the time horizon may result in short run decisions designed only to meet the target rather than designed to maximize the long run conservation goals. Objectives in management for harvested populations typically focus on maximizing the harvest opportunities, while insuring sustainable populations over the time horizon (Caughlan & Oakley 2001; Hauser *et al.* 2007; Nichols *et al.* 2007). Alternatively, the monetary value of the economic yield from harvest might be used (Milner-Gulland 1997; Table 2). Objectives can include both conservation and exploitation of natural resources and can also include several, possibly conflicting, conservation goals. For instance, a conservation problem might deal with the protection of two species that are negatively interacting between one another over an infinite time horizon (Chadès, Curtis & Martin 2012). In metapopulation models, often used in invasion biology, epidemiology and landscape ecology, objectives can also be expressed as maximizing or minimizing the number of sites occupied by a species (Shea & Possingham 2000; Chadès *et al.* 2011; Table 2). When the economic costs of management and monitoring, as well as the cost of failure to maintain a viable protected species are well known, the objective can be clearly formalized to determine the best way to allocate funding to protect a threatened species (Chadès *et al.* 2008; McDonald-Madden *et al.* 2011) or eradicate an invasive species (Regan *et al.* 2006; Baxter & Possingham 2011).

The second step is to define the set of states that represents the possible configuration of the system at each time step. Let $X_t$ be the state variable of the system at time $t$. The state variable can be a population abundance (Milner-Gulland 1997; Runge & Johnson 2002) or predator abundance and prey productivity (Martin *et al.* 2010). Others studies have considered a qualitative state variable such as site occupancy of a colonizing species (Shea & Possingham 2000). We refer to Table 2 for additional examples.

In the third step, one needs to define the decision variable, $A_t$, that is the component of the system dynamic that one can control to meet the objective. For example, it can be expressed as the way of releasing a biocontrol agent in crop sites: many individuals released in few sites or few individuals released in many sites. Another example of control actions is different harvest rates in each age class (Martin *et al.* 2010) or sex class of a species (Milner-Gulland 1997).

The fourth step is to build a transition model describing the system dynamics and its behaviour in terms of the effect of a decision on the state variables (Table 2). This transition model follows a Markov process in which the future state $X_{t+1}$ depends on the current state $X_t$ and the action adopted $A_t$ but not on the past state and action pairs of the system.

In the fifth step, one needs to define the utility function $U_t$ at time $t$ also called the immediate reward. It might be expressed in terms of economic benefits, desired ecological status or social improvement (Table 2) and might be quantified in a more or less subjective way (Simon 1979; Isen, Nygren & Ashby 1988; Milner-Gulland 1997). This function, denoted as $U_t(X_t, A_t)$, which pertains the Markov chain formulation, represents the desirability of acting in a given state of the system and is defined in terms of the state variable $X_t$ (step 2) and the decision $A_t$ (step 3). The utility values can accrue over either a finite or an infinite time horizon depending on the objectives formalized in step 1. In the former case, a terminal reward or salvage value, $R(X_{T+1})$ with $T$ the horizon time, can also be specified that measures the utility that accrues if the system is left in a given state after the last decision is made. In population biology and behavioural ecology, $R(X_{T+1})$ is often chosen to be the desired abundance of a population or the energy state of an individual (Mangel & Clark 1988; Martin *et al.* 2010).

Sixth, the final step consists in determining the optimal solution to the optimization problem. The optimal solution, also called the optimal strategy or policy, maximizes our chance of achieving our objective over a time horizon. An optimal solution is defined as a function $\pi_t{:}X_t \rightarrow A_t$ that maps each state to the optimal action for that state. Hereafter, we examine the three most commonly used approaches to solve an MDP: backward iteration, value iteration and policy iteration.

## How to determine the optimal solution?

Several algorithms using SDP technique are available to find the optimal solution of an MDP. How to choose the most appropriate algorithms mainly depends on the optimization objective (step one). Backward iteration is the run over a finite horizon in time-reversed fashion. It leads to a time- and state-specific optimal solution. Value iteration and policy iteration are used to solve infinite time horizon problems. Both techniques provide an optimal action expressed as a time-independent function.

### OPTIMIZATION PROCEDURE OVER A FINITE HORIZON

According to the principle of optimality (Bellman 1957), an efficient way to find an optimal decision is by reasoning backward in time. More precisely, it consists in assuming that the last decision taken at the horizon time $T$ is optimal and by choosing what to do in every remaining time step. $T$ is the time required to reach the optimal solution. Let $V(X)$ be the value function of states that quantifies the reward or the penalty after each state transition following decision (Lubow 1995). Let $\pi^*$ be a vector that maps the best decision for each state at the horizon time. $\pi^*$ is the set of decisions ($A$) associated with the maximum value function of the set of states [$V(X)$]. Let $\beta$ be the discount factor, representing the value of the reward gained in the next period relative to the reward obtained in the current period (Moore, Hauser & McCarthy 2008; Martin *et al.* 2011). It can also reflect a measure of confidence level in the predictions of the dynamic model. Predictions made for the near future are generally more certain than the ones made for the distant future.

The finite horizon problem can be written formally as

$$V_t(X_t) = \max_{\{A_\tau\}_{\tau=t}^{T}} \sum_{\tau=t}^{T} \beta^{\tau-t} U(X_\tau, A_\tau) + \beta^{T+1} R(X_{T+1}) \qquad \text{eqn 1}$$

The expression includes two parts, the sum of the discounted utility values from time $t$ to the horizon $T$ and the discounted terminal reward ($R(X_{T+1})$), which is a function of the state that the system is left in, $X_{T+1}$ after the last decision is taken.

In the *backward iteration* algorithm, the starting point is to realize that there exists a recursive relationship that identifies, for each state, a value function for step $t$, denoted $V_t(X_t)$, given that step $V_{t+1}(X_{t+1})$ has already been solved (Appendix S1):

$$V_t(X_t) = \max_{A_t}[U_t(X_t, A_t) + \beta V_{t+1}(X_{t+1})] \qquad \text{eqn 2}$$

As suggested by the principle of optimality, the Bellman equation expresses the optimization problem in terms of the current decision alone. The first part of this equation is made of the immediate reward represented by the utility function, while the second part is the value function for the next period, $V_{t+1}(X_{t+1})$. The procedure is initialized by setting $V_{T+1}(X_{T+1}) = R(X_{T+1})$. Then, the previous value $V_T(X_T)$ is computed, then $V_{T-1}(X_{T-1})$, and so on. The optimal action, that is the action associated with each initial state $X_0$, is obtained by repeated backward recursions from the horizon time $T$ to present time 0 (see Fig. 1b–d) and by taking the argument of the maximum initial values $V_0(X_0)$ (Fig. 1d and Fig. 2).

An important issue, besides the choice of the horizon $T$ and of the terminal value of the system, $R(X_{T+1})$, is the choice of a discount factor β (Lubow 2001) which lies between 0 and 1 (Bellman 1957). Discounting is often specified in terms of a discount rate $r$, with the (annual) discount factor given by $\beta = 1/(1 + r)$ Conservation biologists are more likely to use a β of 1, meaning the value of future system states is not discounted. In such situations, future utility contributes as much to the overall objective as current utility.

Even though not discounting future utility complies with the sustainability principle, most economists recommend using a discount factor <1. One reason is that many people place more importance on current than future rewards, especially when future rewards are risky (Norgaard & Howarth 1991). In addition, most problems in resource management involve utilities that have some social and economic cost and benefit, associated with them. When the resource has a non-market value, it can be difficult to convert the ecological, social and economic costs and benefits into a common scale (Wam 2009). Such scale differences and issues of utility incommensurability impede the determination of an appropriate discount rate (whether financial, social or ecological). The method commonly used for selecting a discount rate is based on a market rate for a

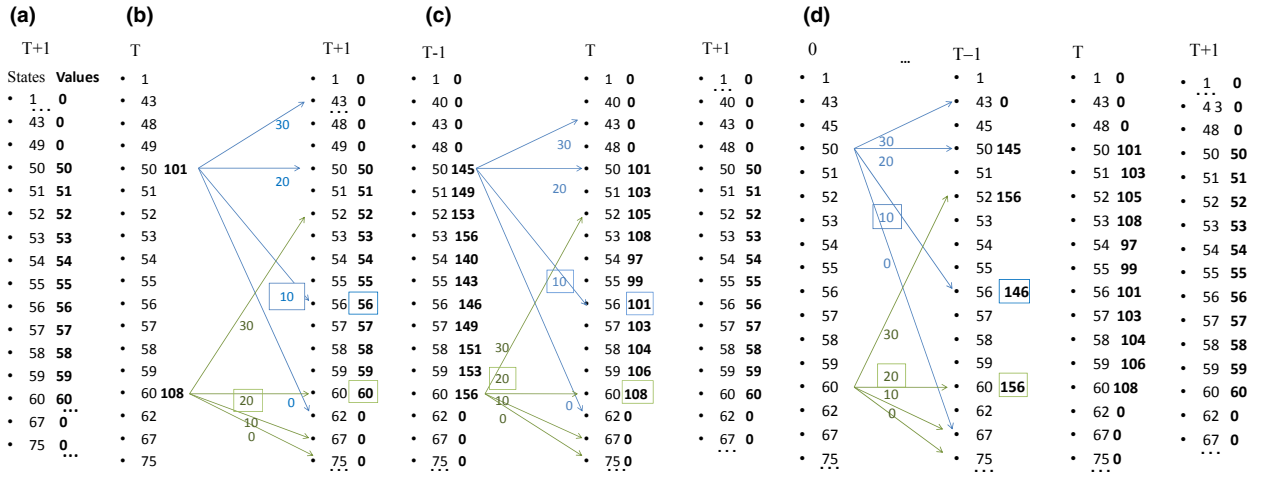


**Fig. 1.** The backward iteration showing successive transition states and the best harvest strategy (in%). (a) First step shows all realizations of state variable $X$ standing for population size, varying from 1 to 250 individuals and the associated values of the terminal reward $R(X_{T+1})$ (bold column). For convenience, we shorten the objective to maintaining an abundance range of $N_{min} = 50$ and $N_{max} = 60$ so that the values at the endpoint T + 1 worth 0 from state 1 to state $N_{min}-1$ and then take the value of the states from $N_{min}$ to $N_{max}$. Beyond $N_{max}$, values are again set to 0. (b) We proceed backward in time and define possible realizations for states at time $T$. In this example the state space remains the same across the horizon time. Here we only looked at four potential actions: do nothing, harvest 10%, 20% and 30% of the population. The arrows illustrate the deterministic dynamic of the system, and represent the exponential growth from 1 year to another (with λ=1.25), the transition states given the harvest strategies. The best strategy is framed in blue (for $N_{min}$) and in green (for $N_{max}$), and it stands for the action that maximizes the values at $T$+1 also framed in colored squares. For instance with a population of 50 individuals (blue), among the four possible state transitions, the action associated to the highest terminal value is a 10% harvest. So this action is optimal and allows the calculation of the value function of the blue state at time T: $V(50)_T = 50*(1-0.10) + 56 = 101$. We proceed the same way for all possible states at time T + 1, in order to fill the vector of the value function. This allows us to determine the optimal action for each state at T that is associated to the maximum value function at time T + 1. (c) At T−1, we look again at the transition states for $N_{min}$ and $N_{max}$ given the potential actions and we choose the strategy that leads to the next state (at $T$) showing the highest value. At this step, there are two actions associated to the maximum value function V(50) = 101. In our example, we picked the minimum harvest rate when there were several optimal actions. (d) At time 0, we look one last time at the transition states for $N_{min}$ and $N_{max}$ given the potential actions and the strategy that leads to the highest value that is then the optimal solution. So in the backward iteration, optimal action is reached when the procedure reaches the initial time 0.

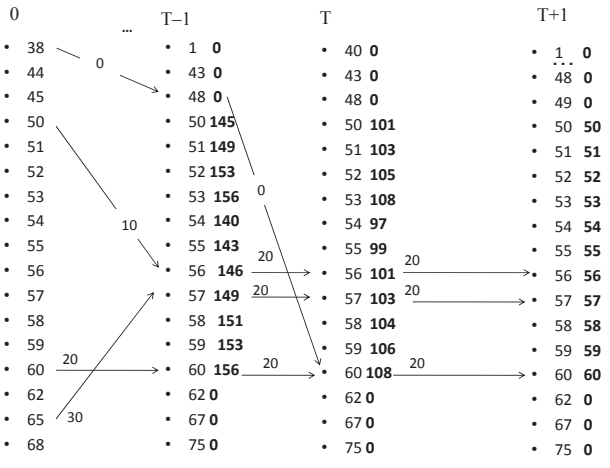


**Fig. 2.** End of the backward iteration and time sequence of actions for four initial states. Here, we display the time sequence of actions standing for the trajectory that leads from any initial state to the final objective that is to keep a population between $N_{\min}$ and $N_{\max}$ while harvesting as few individuals as possible. In the example, the recursion starts from the end and goes backward in time. Once values of all states are obtained across the time horizon, we can decide which actions to take across the successive transition states. Here, we look at the best trajectory for four initial states $N = 38$, 50, 60 and 65 individuals.

relatively risk-free asset such as a US Treasury bond. Recent recommendations for environmental projects suggest the use of $r = 2\%$ for long-term projects (http://www.whitehouse.gov/omb/circulars_a094/a94_appx-c; see also EU's 'Guide to Cost-Benefit Analysis of Investment Projects').

### OPTIMIZATION PROCEDURE OVER AN INFINITE HORIZON

With infinite horizon problems, both the value function and the optimal policy are independent of time. The problem to be solved can be written as

$$V(X_t) = \max_{A(X)} \sum_{\tau=0}^{\infty} \beta^{\tau-t} U(X_\tau, A(X_\tau)) \qquad \text{eqn 3}$$

Starting with an arbitrary value function and iterating over an infinite horizon model with policy or value iteration causes the optimal action to converge towards a time-independent function also called a stationary strategy with the optimal solution only depending on the state of the system and not on time.

The first algorithm used to solve an MDP over an infinite horizon, called *value iteration*, follows the same procedure as described previously except that the Bellman equation is applied iteratively until a convergence criterion is met. A typical convergence criterion (Boutilier, Dearden & Goldszmidt 2001) is

$$\|V(X_{t+1}) - V(X_t)\| \leq \frac{\varepsilon(1-\beta)}{2\beta} \qquad \text{eqn 4}$$

where the norm $\|V(X_{t+1}) - V(X_t)\|$ is the maximum absolute value of the difference between two successive decision values, for all possible states. The value of $\varepsilon$ is usually chosen to be small, so that when the condition in eqn 4 is satisfied, the value function is within $\varepsilon$ of its optimal value. In our example, we fixed $\varepsilon$ at $10^{-3}$ as in Boutilier, Dearden & Goldszmidt (2001).

Another algorithm called *policy iteration* (Howard 1960) involves alternating between finding the best policy (or strategy) given the current guess of the value function and determining the value function associated with the current policy (Appendix S2). One advantage of the policy iteration algorithm is that it will generally run faster than the value iteration (Howard 1960). The policy iteration approach can be decomposed in two steps.

In the first step (*evaluation*), a value function is calculated from a guessed policy (Boutilier, Dearden & Goldszmidt 2001). Let $A_t$ be any policy which describes the actions that are taken for any value of the state $X_t$, so that $X_{t+1}$ is a function of both the state and action variables that can be written as $X_{t+1} = g(X_t, A_t)$. The value function associated with this policy can be determined by solving a system of linear equations, one for each value of the state variable

$$V_t(X_t) = U_t(X_t, A_t) + \beta V_{t+1}(g(X_t, A_t)) \qquad \text{eqn 5}$$

In the second step (*improvement*), we find the policy $A'$ that satisfies, for each value of the state

$$\max_{A'} U(X_t, A'_t) + \beta V_{t+1}(g(X_t, A_t)) \qquad \text{eqn 6}$$

The same procedure is performed again (back to first step) until the two policies $A$ and $A'$ do not change.

In infinite time horizon problems, the standard approaches may need to be modified if a discount rate of 0 is used. With no discounting of the future, the value function will not be stationary unless there is a probability of 1 that the state variable reaches and stays in a non-valued state at some time, such as extinction in a population conservation problem. If there is a positive probability of obtaining a positive reward in any given future period, the expected value of future rewards will be infinite, and hence, V will not well defined. In this situation, it is therefore more appropriate to use an average value approach that attempts to maximize the per period expected value function. Algorithmically, there are two main approaches to solve such problems. The vanishing discount approach uses a discount factor near 1 (such as 0.999999). The relative value approach solves for the average reward plus an adjustment to account for the relative value of being in alternative states. For further discussion, see Puterman (1994).

## Making decisions in presence of uncertainty

Thus far, we have focused on deterministic MDPs in which each state and action combination yields a unique, known result. Here, we discuss how to accommodate uncertainty in dynamic programming. In SDP, there are several possible next states, given the action taken and the current state and each of them has a certain probability to be achieved. Let $P$ be a transition matrix displaying the conditional probabilities of the system at state $X_t$ at time $t$ and action $A_t$ (in rows) to change into

states $X_{t+1}$ (in columns) given the action. The transition matrix is a stochastic matrix which consists of non-negative elements with rows that sum to 1. The Bellman equation can be rewritten as the sum of the utility value at the current state (which holds in the deterministic version) and the sum of the expected future rewards that are the products of transition probabilities and values of all possible next states (Appendix S1). In the backward iteration procedure, for example, the stochastic version of the equation is

$$V_t(X_t) = \max_{A_t}\left[U_t(X_t, A_t) + \beta \sum_{X_{t+1}} P(X_{t+1} \mid X_t, A_t) V_{t+1}(X_{t+1})\right] \quad \text{eqn 7}$$

One may notice that the difference from eqn 2 is the addition of the transition probability matrix. Actually, the deterministic version of the Bellman equation can be rewritten as a special case of SDP, where $P$ is a matrix of 0s with a single 1 in each row. In SDP, $P$ consists of transition probabilities depending on stochastic events related to demographic and/or environmental stochasticity or to the action taken, the effect of which can be uncertain.

We distinguish several types of uncertainty that can be accounted for to solve a Markov decision problem. First, there is the natural uncertainty which is related to natural and inherent processes occurring in the system and its environment. It is difficult to measure and even more difficult, if not impossible, to reduce. Populations are subjected to environmental stochasticity that can strongly affect their vital rates through changes in weather conditions, habitat structure or other external biotic and abiotic factors (Regan, Colyvan & Burgman 2002; Martin *et al*. 2010). Demographic stochasticity is also a common source of natural uncertainty. It reflects the variability in survival and reproduction among individuals and is likely to occur in small-size populations (Lande 1993).

Second, management uncertainty, also called partial controllability, results from the inability to accurately apply the decision being made (Williams 2011). Sometime, actions themselves are taken in an uncertain way. For instance, a planned harvest rate or a prescribed burn can sometimes not be achieved by wildlife or forest managers for many reasons even though it was assumed to be the best solution (Milner-Gulland 1997; Baxter & Possingham 2011; Richards, Possingham & Tizard 1999; see also Table 3).

The third type of uncertainty deals with that coming from the partial knowledge of the value of the state variable. To cope with such uncertainty, one may use partially observable Markov decision process (POMDP), a procedure that can solve stochastic dynamic problems assuming we are unable to observe perfectly the state of the system (Chadès *et al*. 2008). In a population model, a POMDP might augment an MDP to include detection probability matrices. The detection history is not explicitly represented but rather is summarized by a belief state or probability distribution over the state space representing where we think the state of the system is (Chadès *et al*. 2008; see also Table 3). Unfortunately, POMDPs are even more complex to solve than MDPs, and to date, it is possible to compute exact solutions only for small-size problems (Chadès *et al*. 2011).

**Table 3.** Main features of software packages implementing dynamic programming. MDPSolve (https://sites.google.com/site/mdpsolve/) and MDPToolbox (http://www.inra.fr/mia/T/MDPtoolbox/) are considered. MDP is for Markov decision process, POMDP for partially observed Markov decision process and AM for adaptive management.

| | MDPSolve | MDPtoolbox |
|---|---|---|
| Natural and Management uncertainty | Yes (infinite/finite); Value, policy, backward iteration | Yes (infinite/finite); Value, policy, backward iteration |
| *Comments* | f2p and g2p functions compute the transition matrix | Need to build transition matrix |
| Observation uncertainty (POMDP) | Yes | No |
| *Comments* | Infinite or finite horizon | |
| Model uncertainty and Parametric uncertainty (AM) | Yes | No |
| *Comments* | Passive and Active | Passive AM to be included in future release |
| Unknown uncertainty (reinforcement learning) | No | Yes (on Infinite horizon) |
| *Comments* | | Q-learning algorithm |

Another form of uncertainty is model uncertainty, which refers to the lack of certainty about the structural frame shaping the behaviour of the system (Walters 1986; Punt & Hilborn 1997). Adaptive Management is a common approach adopted to reduce such uncertainty by testing multiple models through the ongoing process of management and monitoring occurring under the principle of 'learning by doing' (Runge 2011). In adaptive management, belief weights are attributed to each model depending on the comparison between model predictions of the outcome of an action and the observed response from monitoring. Such a comparison allows us to increase our belief in the model that is most likely to give rise to the observed response.

Two approaches, based on the role of learning, are then conceivable (Williams 2009). Passive adaptive management assumes learning is a by-product of decision-making in which the models weights are updated by applying Bayes theorem but remain constant during the optimization process (Williams, Nichols & Conroy 2002). For instance, Martin *et al*. (2010) used passive adaptive management to determine an optimal harvest strategy to control raccoons to improve oystercatcher productivity. They considered two models, one assuming no effect of raccoons on oystercatchers' productivity and another one assuming a strong effect. In the second approach, referred to as active adaptive management, model

weights appear in the optimization process. More precisely, the next updated weights are incorporated in the expected sum of future rewards of the Bellman equation. Such approach is the most advanced form of adaptive management. In contrast to passive adaptive management, active adaptive management considers how current decisions will affect future learning and chooses an optimal balance between rewards based on current beliefs and future rewards based on updated beliefs (Runge 2011). For instance, McDonald-Madden *et al.* (2011) used active adaptive management to assess species relocation strategies in the context of climate change. They considered two models, one in which carrying capacity declined over time because of climate change and another one in which climate change had no impact on species carrying capacity.

The last form of uncertainty, referred to as parametric uncertainty, is related to our limited knowledge about the parameters that govern the system dynamics (Williams 2009). This optimization problem is also referred to as adaptive management under parameter uncertainty. One approach to this problem makes use of conjugate priors over the unknown parameters (Raiffa & Schaifer 1961). For example, Walters & Hilborn (1976) used a normal prior over parameters in population growth model. Hauser & Possingham (2008) and Rout, Hauser & Possingham (2009) used Beta priors to represent uncertainty over transition probabilities. Recently, approximate approaches using projection methods have been developed for situations that do not support the use of conjugate priors. Springborn & Sanchirico (2013) applied this approach to the management of development that impacts mangrove habitat.

When the form of uncertainty is unknown, an alternative optimization approach to backward iteration, policy or value iteration is reinforcement learning. This technique makes sequential decisions when transition probabilities or rewards are unknown and cannot be estimated by simulation (Chadès, Curtis & Martin 2012). The Q-learning algorithm is used in which the optimal value $V_0$* and the corresponding action are estimated by a learning process of observed transitions and values obtained with function approximation (Chadès *et al.* 2007; Table 3). A potential issue with this method, originally developed in robotics, is that it requires a large number of observations to build the transition matrix.

## Software packages performing dynamic programming

There are several software packages that allow the implementation of SDP. Adaptive Stochastic Dynamic Programming (ASDP) Lubow 1995, was the first application developed for biologists to solve optimization problems using dynamic programming. It is a MS DOS executable that is no longer maintained by its author. Two other packages are available for MATLAB: MDPSolve (Fackler 2011, available at https://sites.google.com/site/mdpsolve/) and MDPtoolbox (version 4.0) available at http://www.inra.fr/mia/T/MDPtoolbox/. MDPtoolbox is also available for the open-source software for numerical computations Scilab, R (http://cran.r-project.org/web/packages/MDPtoolbox) and GNU Octave (GNU's not Unix). Both MDPSolve and the MDPtoolbox implement the value iteration and the policy iteration algorithms, while ASDP uses only the former. Adaptive Stochastic Dynamic Programming does not use the convergence criterion discussed previously for infinite time horizon but stops after the policy remains the same for a specified number of iterations. MDPSolve and MDPtoolbox deal with natural and management uncertainty in finite and infinite time horizons (Table 3). MDPtoolbox satisfyingly copes with unknown management uncertainty through the implementation of Q-learning in an infinite horizon, while MDPSolve does not. MDPSolve enjoys capabilities that permit solving POMDP and addressing model uncertainty, while MDPtoolbox does not. Transitions for continuous variables are often defined in terms of either a conditional density or a transition equation which specifies the next period state as a function of the current state and, possibly, a random shock reflecting environmental variation or other process noise. MDPSolve has procedures that define discrete transition matrices that approximate the transition for continuous variables.

In the following section, we provide an application of SDP and solve the associated decision problem using both MDPSolve and MDPtoolbox. Although we emphasize that this exercise does not represent a general introduction to these packages (we refer to the user's guides instead), we hope it will be a good starting point. In addition to the use of these packages, we demonstrate that MDP problems can be implemented in program R and provide code that can be amended for one's own purpose.

## Application to wolf culling

In this section, we illustrate each step of SDP required to derive an optimal management strategy to control a population of wolves in Europe. We consider several decision models of increasing complexity for wolf culling. First, we build a deterministic model to keep things easy and illustrate the notation. Then, we illustrate how to make decisions when uncertainty exists.

### SETTING THE SCENE

We go through the six steps of dynamic programming. First, the optimization objective is to maximize the population while providing that the population does not exceed 250 individuals ($N_{max}$) and remains above 50 individuals ($N_{min}$). These thresholds are somewhat arbitrary from a biological perspective, but were selected to obtain results in a reasonable amount of time while scanning a relatively broad range of abundance states. Second, the state variable $X_t$ is the population size $N_t$ at time $t$, which ranges from 0 to $K$ where $K$ is an arbitrary upper bound on the state space. Third, the control variable $A_t$ is the harvest rate $H_t$, a discrete variable ranging from 0% to 100% with an increment of 1/(K + 1) therefore allowing as many possible actions as there are number of states. Fourth, regarding the transition model describing population dynamics and the

 

consequences of actions (harvest $H_t$) on the state variable (abundance $N_t$), we adopted an exponential growth (Fig 1), which is suitable to describe a population currently in a colonization phase. We assumed an additive effect of human offtake on total mortality (Creel & Rotella 2010; Murray *et al.* 2010). More precisely, we used:

$$N_{t+1} = \lambda N_t(1 - H_t) \quad \text{eqn 8}$$

where λ is the population growth rate. The value of λ was extracted from the literature using the French population as an example (the estimate of λ is 1.25 with 95% confidence interval [1.14; 1.37]; Marescot *et al.* 2011). Fifth, utility is based on abundance and harvest rate bearing in mind the objective to keep a population size between $N_{\min}$ and $N_{\max}$. We choose a utility function that is increasing linearly with abundance when the current state is within the objective range. In mathematical terms, we write:

$$U_t = N_t(1 - H_t)\alpha_t \quad \text{eqn 9}$$

where $\alpha_t$ takes the value 1 if $N_{\min} \leq N_{t+1} \leq N_{\max}$ and 0 otherwise. Given the current population size $N_t$ and harvest rate $H_t$, if the future state is above the utility threshold $N_{\max}$ or below $N_{\min}$, the penalty factor $\alpha_t$ takes a null value and, therefore, the utility function does as well. If, however, the future population size $N_{t+1}$ is in the target abundance range, then the utility of harvest level $H_t$ in state $N_t$ is the population size after harvest but before annual growth occurs (Fig 1b). Because we assumed exponential growth, and because $N_{\max}$ is below the carrying capacity and the growth rate is greater than 1, the objective can be translated into attaining and then maintaining the population at $N_{\max}$. An alternative utility function could be defined only on the current abundance because no economic cost was considered here. Adopting the general formulation in which utility is defined as a function of current action would be useful to incorporate economic costs and pay-offs. Sixth, we need to solve the Bellman equation using the value iteration or the policy iteration algorithm.

### DETERMINISTIC CASE

We first ran a deterministic model over an infinite time horizon using both value iteration and policy iteration algorithms. There was also an analytic solution to this deterministic MDP, which enables us to validate the approach. With an objective of keeping a population between $N_{\min}$ and $N_{\max}$, the optimal action for a state $N$ is a harvest rate of the maximum between 0 and $1 - N_{\max}/(\lambda N)$ which removes the exact surplus of individuals above $N_{\max}$ as in our linear utility function. The three different methods provided the same optimal harvest rates. The strategy of no culling remained the best strategy until population reached 200 individuals. Above 200 individuals, expected population size reached the utility threshold $N_{\max}$ ($200 \times 1.25 = 250$). From there, optimal harvest rate increased from 0.8% to 20%. The highest harvest rate was reached at the utility abundance threshold of 250 individuals. We provide R code to implement the resolution of this MDP (Appendices S1 and S2). This example was also run in MDPSolve and MDPtoolbox (Appendix S3 for the scripts and S4 for the numeric values).

The solution demonstrates the trade-off between current and future utility inherent in dynamic programming problems. Here, there is no reason to cull unless the population will exceed $N_{\max}$ in the next period. If the population is high enough, however, it is optimal to forgo current utility by culling enough to ensure that utility is obtained in the next period.

### COPING WITH UNCERTAINTY

Besides the deterministic model, we consider models with demographic stochasticity that generates variability in population growth rates arising from random differences among individuals in survival and reproduction within a season or a year (Lande 1993). R code is provided to run this additional example (Appendix S5).

We assume that the state variable is distributed according to a Poisson distribution:

$$N_{t+1} \sim Pois(\lambda N_t(1 - H_t)) \quad \text{eqn 10}$$

with mean value $E(N_{t+1}) = \lambda N_t(1 - H_t)$ equal to its deterministic counterpart (Appendix S1 and S2). The transition probabilities are now changing across the different states according to a Poisson distribution:

$$P(X_{t+1} = n_{t+1} | X_t = n_t, a_t = h_t) = e^{(-\lambda n_t(1-h_t))} \frac{[\lambda n_t(1 - h_t)]^{n_{t+1}}}{n_{t+1}!} \quad \text{eqn 11}$$

We found that harvesting was not recommended as long as population was below 200 individuals. As in the deterministic model, above this abundance threshold, harvesting increased from 0.8% to 20% of population size (Fig. 3). When population was already at the upper objective limit $N_{\max}$, 50 individuals were to be removed.

## Discussion

Stochastic dynamic programming is a valuable tool for solving complex decision-making problems, which has numerous applications in conservation biology, behavioural ecology, forestry and fisheries sciences. It provides an optimal decision that is most likely to fulfil an objective despite the various sources of uncertainty impeding the study of natural biological systems. The formalization of objectives of any Markov decision problem is given by the utility function that allows prioritizing the preferences of the ones who make the decisions (decision-maker or manager). As opposed to the dynamic model, the representation of utility is subjective and hence can be difficult to define.

### DIFFERENT WAY OF DEFINING A UTILITY FUNCTION

The use of dynamic programming implies a particular formalization of the objective into a utility function. The utility is a

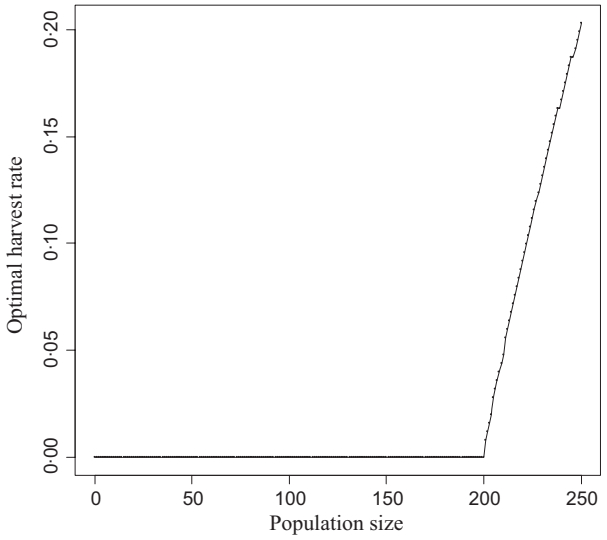


**Fig. 3.** Optimal harvest obtained from a model incorporating demographic stochasticity. Demographic stochasticity stands for individual variability in vital rates. The stochastic dynamic programming was run over a finite time horizon (150 years) with the backward iteration procedure. The transition probabilities are changing across the different states according to a specific density function, here that of a Poisson distribution of parameter the next population states.

function of one or more decision variables, themselves defined on the system states and actions. Utility is a sometimes defined with constraints that can reflect different decision rules (Williams, Nichols & Conroy 2002).

Problems in resource management often deal with trade-offs not only between current and future objectives but also between multiple current objectives. For example, one trade-off objective is to control and protect a predator that is potentially threatened. Other objectives can be to restore natural habitat while minimizing action cost and allowing some recreational activities. When multiple objectives are involved, different decision variables can be considered. The objective can be to find a relevant utility optimum reflecting the trade-off between the different decision variables (for instance, the habitat quality and the intensity of recreational activity) which can respond differently to decisions (restoration). In such cases, some weighting scheme must be used to express the different decision variables in common units. For example, suppose that $E$ is an environmental performance of variable and $B$ is the benefits from recreation activities and $C$ is the financial cost of an action. Utility can be defined as a weighted sum of the decision variables $U = wE + B - C$, where $w$ is a weight that assigns a monetary value to environmental variable.

An alternative to using weighted sums is to use a multiplicative functional form such as $U = E^a B^b$. The parameters $a$ and $b$ serve two functions. First, if $a$ and $b$ are both positive and if $a > b$, it implies that environmental variable is more important than the recreation variable. The relative value of $a$ to $b$ changes the weight that is placed on $E$ versus $B$. Second, if $a$ or $b$ value is <1 in absolute value, it implies that the marginal contribution of an additional unit is smaller for larger values of the variable than for smaller one. This representation is also appropriate when it is deemed more important to save an additional individual of a protected species such as the wolf in France when there are very few remaining than when the population is more abundant. Note that unlike the additive utility form, this multiplicative form is not affected by the scale of either variable.

Another approach is to convert one decision variable into a constraint or to use a penalty function for failure to meet the target. This approach simplifies the multiple objectives into a single constrained objective (Converse *et al.* 2012). For instance, one objective can be to improve habitat quality given a limited budget of \$50 000, while allowing a minimum of 100 h/year of recreational activities. For example, $U = E$ if ($B \geq 100$ h/year) and if ($C <$ \$50 000); otherwise $U = 0$. Here, the decision variable is the intensity of recreation, and action cost has been converted into a constraint. This avoids the need to make comparisons between variables of different types, but it also has implications that an analyst should be aware of. First, if the system never reaches the threshold implied by the two constraints (100h/year of recreation and a budget of \$50 000), it means that both $B$ and $C$ are irrelevant. Second, it implies that once one threshold is reached, further increases in $C$ or further decreases in $B$ are irrelevant. Finally, it should be noted that this utility is not the same as optimizing with respect to $E$ subject to a long run expectation that the thresholds are satisfied.

## LIMITS OF DYNAMIC PROGRAMMING: CURSE OF DIMENSIONALITY

Despite the flexibility of dynamic programming, one has to find a trade-off between biological realism and model complexity when tackling an optimization problem. Indeed, DP methods often face the issue known as 'the curse of dimensionality' which states that, when more state variables are added in the model, the size of the DP problem increases exponentially (Walters & Hilborn 1978; Schapaugh & Tyre 2012). To overcome this computational complexity, approximate optimization methods can be used such as heuristic sampling algorithms that proved efficient for models with several variables (Nicol & Chadès 2011). These methods approximate the optimal solution given the starting state by simulating the possible future states the more likely to occur. Simulating only possible future states lightens the computational calculation in comparison with the value or policy iteration procedure in which values are computed for all possible states.

## PERSPECTIVES FOR WOLF POPULATION MANAGEMENT

The aim of this study was to demonstrate the usefulness and relative ease of SDP. We hope that this study can serve as an entry point into the extensive literature and potential applications of SDP in ecology. For the sake of clarity, we made assumptions to keep the illustration simple, but SDP can accommodate several useful extensions. For example, we did not include socio-economic constraints in the modelling

process. However, SDP allows the incorporation of such factors by maximizing several objectives simultaneously using complex trade-offs in the utility function (Walters & Hilborn 1978; Milner-Gulland 1997; Runge & Johnson 2002). In our example, economic constraints could be incorporated via a trade-off between monetary loss from livestock depredation, impact of wolves on game abundance and indirectly on hunting activity, the receipts from ecotourism and the cost of wolf culling (e.g. Milner-Gulland 1997). Second, the lower abundance limit could also be refined based on an ecological threshold that once reached is irreversible (Holling 1973; Bodin & Wiman 2007). Using such thresholds would be relevant for a protected species because it would insure population viability without necessarily changing the optimal policy (Martin *et al*. 2009). Additionally, further work is needed to compare optimal strategies obtained with alternative population dynamic models. Indeed, the choice of exponential growth is an adequate model for a colonizing population, but when a population is established and the habitat saturated, this model becomes inappropriate. Instead of considering exponential growth, one could use a logistic growth with density-dependent effects such as an Allee effect which has been shown in social species with few breeding units like African wild dogs (*Lycaon pictus*) (Stephens & Sutherland 1999).

### PERSPECTIVES FOR ADAPTIVE MANAGEMENT

Structural uncertainty can be defined as the noise arising from our lack of knowledge about system behaviour and can be reduced through comparison of multiple models (Walters 1986; Punt & Hilborn 1997; Williams, Nichols & Conroy 2002; Dorazio & Johnson 2003). For example, one could also assess the impact of accounting versus neglecting poaching on the final optimal action (Milner-Gulland 1997) or the impact of additive versus compensatory effects of harvesting on annual mortality (Runge & Johnson 2002). Reducing structural uncertainty is essential for conducting a conservation programme, especially when the resulting optimal policy is highly sensitive to models structure and assumptions. In such case, one needs the most accurate predictions to optimize future allocation of monitoring and management effort (Williams, Nichols & Conroy 2002; Conroy *et al*. 2008). Adaptive management is a sequential action process, specifically designed for conservation problems dealing with structural uncertainty (Runge 2011). It is an integrated part of decision-making that deals simultaneously with predictions on future states and updated beliefs from monitoring (Walters 1986). Using SDP in an adaptive management framework aims at seeking the optimal management strategy while reducing structural uncertainty, so better knowledge leads to better actions (Martin *et al*. 2009). However, the real interest of adaptive management in conservation biology is not really to reduce structural uncertainty that sometimes does not affect the optimal solution but more to drive a learning process to improve decision given management objectives (Runge 2011).

One common assumption in conservation biology is that a system must be well understood before making any management decision. Monitoring efforts tend to be oriented towards the perspective of understanding system functions more than towards the establishment of good decision rules. This leads sometimes to inefficient outlays of conservation funds (Caughlan & Oakley 2001; Field, Tyre & Possingham 2005; Chadès *et al*. 2008). Considering the environmental issues currently at stake, we agree with Nichols & Williams (2006) that active conservation action should be initiated even when the causes of the problem are not fully identified. Stochastic dynamic programming is a relevant optimization method for making decisions while conducting monitoring. Biologists studying ecological systems are often facing uncertainty, noise and disturbance. Adaptive management is a further natural extension of SDP and should be the preferred approach undertaken whenever a management action is planned.

## Acknowledgements

This research was conducted with the support of funding from the Australian Government's National Environmental Research Program and an Australian Research Council's Centre of Excellence (IC). We thank Tara Martin for comments on the manuscript and all the organizers of the workshop on optimization methods in natural resources management in June 2011 Raleigh, NC.

## References

Baxter, P.W.J. & Possingham, H.P. (2011) Optimizing search strategies for invasive pests: learn before you leap. *Journal of Applied Ecology*, **48**, 86–95.

Bellman, R. (1957) *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, USA.

Bodin, P. & Wiman, B.L.B. (2007) The usefulness of stability concepts in forest management when coping with increasing climate uncertainties. *Forest Ecology and Management*, **242**, 541–552.

Boutilier, C., Dearden, R. & Goldszmidt, M. (2001) Stochastic dynamic programming with factored representations. *Artificial Intelligence*, **121**, 49–107.

Caughlan, L. & Oakley, K.L. (2001) Cost considerations for long-term ecological monitoring. *Ecological Indicators*, **1**, 123–134.

Chadès, I., Curtis, J.M.R. & Martin, T.G. (2012) Setting realistic recovery targets for interacting endangered species. *Conservation Biology*, **26**, 1016–1025.

Chadès, I., Martin, T. G., Curtis, J. M. & Barreto, C. (2007) Managing interacting threatened species: a reinforcement learning decision theoretic approach. *MODSIM 2007 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, (eds L. Oxley & D. Kulasiri), pp. 74–80. Modelling and Simulation Society of Australia and New Zealand, Christchurch, New-Zealand.

Chadès, I., McDonald-Madden, E., McCarthy, M.A., Wintle, B., Linkie, M. & Possingham, H.P. (2008) When to stop managing or surveying cryptic threatened species? *Proceedings of the National Academy of Sciences*, **105**, 13936–13940.

Chadès, I., Martin, T.G., Nicol, S., Burgman, M.A., Possingham, H.P. & Buckley, Y.M. (2011) General rules for managing and surveying networks of pests, diseases, and endangered species. *Proceedings of the National Academy of Sciences*, **20**, 8323–8328.

Clark, C.W. & Mangel, M. (2000) *Dynamic State Variable Models in Ecology: Methods and Applications*. Oxford University Press, New York, USA.

Conroy, M.J., Barker, R.J., Dillingham, P.W., Fletcher, D., Gormley, A.M. & Westbrooke, I.M. (2008) Application of decision theory to conservation management: recovery of Hector's dolphin. *Wildlife Research*, **35**, 93–102.

Converse, S.J., Moore, C.T., Folk, M.J. & Runge, M.C. (2012) A matter of tradeoffs: reintroduction as a multiple objective decision. *The Journal of Wildlife Management*, **78**, 124–136.

Creel, S. & Rotella, J.J. (2010) Meta-Analysis of relationships between human offtake, total mortality and population dynamics of gray wolves (*Canis lupus*). *PLoS ONE*, **5**, 12918.

Dorazio, R.M. & Johnson, F.A. (2003) Bayesian inference and decision theory – a framework for decision making in natural resource management. *Ecological Applications*, **13**, 556–563.

European Commission Environment (1992). *Council Directive 92/43/EEC of 21 May 1992 on the conservation of natural habitats and of wild fauna and flora*. European Commission Environment, Brussels, Belgium. http://ec.europa.eu/environment/nature/legislation/habitatsdirective/index_en.html

Fackler, P. (2011) MDPSOLVE a Matlab Toolbox for Solving Markov Decision Problems with Dynamic Programming. User's Guide.

Field, S.A., Tyre, A.J. & Possingham, H.P. (2005) Optimizing allocation of monitoring effort under economic and observational constraints. *Journal of Wildlife Management*, **69**, 473–482.

Hauser, C.E. & Possingham, H.P. (2008) Experimental or precautionary? Adaptive management over a range of time horizons. *Journal of Applied Ecology*, **45**, 72–81.

Hauser, C.E., Runge, M.C., Cooch, E.G., Johnson, F.A. & Harvey, W.F. IV (2007) Optimal control of Atlantic population Canada geese. *Ecological Modelling*, **201**, 27–36.

Holling, C.S. (1973) Resilience and stability of ecological systems. *Annual Review of Ecology and Systematics*, **4**, 1–23.

Houston, A., Clark, C., McNamara, J.M. & Mangel, M. (1988) Dynamic models in behavioral and evolutionary ecology. *Nature*, **332**, 29–34.

Howard, R.A. (1960) *Dynamic Programming and Markov Processes*. Technology Press of Massachusetts Institute of Technology, Cambridge, MA.

Intriligator, M.D. (1971) *Mathematical Optimization and Economic Theory*. Prentice-Hall, Englewood Cliffs, New Jersey, USA.

Isen, A.M., Nygren, T.E. & Ashby, F.G. (1988) Influence of positive affect on the subjective utility of gains and losses: it is just not worth the risk. *Journal of Personality and Social Psychology*, **55**, 710–717.

Johnson, F.A., Moore, C.T., Kendall, W.L., Dubovsky, J.A., Caithamer, D.F., Kelley, J.R. & Williams, B.K. (1997) Uncertainty and the management of mallard harvests. *Journal of Wildlife Management*, **61**, 202–216.

Lande, R. (1993) Risk of population extinction from demographic and environmental stochasticity and random catastrophes. *American Naturalist*, **142**, 911–927.

Lembersky, M.R. & Johnson, K.N. (1975) Optimal policies for managed stands: an infinite horizon Markov decision process approach. *Forest Science*, **21**, 109–122.

Lubow, B.C. (1995) SDP: generalized software for solving stochastic dynamic optimization problems. *Wildlife Society Bulletin*, **23**, 738–742.

Lubow, B.C. (2001) *Adaptive Stochastic Dynamic Programming (ASDP): Version 3.2*. Colorado State University, Fort Collins, Colorado, USA.

Ludwig, D. & Rowe, L. (1990) Life history strategies for energy gain and predator avoidance under time constraints. *American Naturalist*, **135**, 686–707.

Mangel, M. & Clark, C.W. (1988) *Dynamic Modeling in Behavioral Ecology*. Princeton University Press, Princeton, New Jersey, USA.

Marescot, L., Pradel, R., Duchamp, C., Cubaynes, S., Marboutin, E., Choquet, R., Miquel, C. & Gimenez, O. (2011) Capture-recapture population growth rate as a robust tool against detection heterogeneity for population management. *Ecological Applications*, **21**, 2898–2907.

Martin, T.G., Chadès, I., Arcese, P., Marra, P.P., Possingham, H.P. & Norris, D.R. (2007) Optimal conservation of migratory species. *PLoS ONE*, **2**, e751. doi:10.1371/journal.pone.0000751.

Martin, J., Runge, M.C., Nichols, J.D., Lubow, B.C. & Kendall, W.L. (2009) Structured decision making as a conceptual framework to identify thresholds for conservation and management. *Ecological Applications*, **19**, 1079–1090.

Martin, J., Kendall, W.L., O'Connell, A.F., Simons, T.R., Waldstein, A.H., Smith, G.W. *et al.* (2010) Optimal control of native predators. *Biological Conservation*, **143**, 1751–1758.

Martin, J., Fackler, P.L., Nichols, J.D., Lubow, B.C., Mitchell, J., Eaton, M.J., Runge, M.C., Stith, B.M. & Langtimm, C.A. (2011) Structured decision making as a proactive approach to dealing with sea level rise in Florida. *Climatic Change*, **107**, 185–202.

McDonald-Madden, E., Runge, M.C., Possingham, H.P. & Martin, T.G. (2011) Optimal timing for managed relocation of species faced with climate change. *Nature Climate Change*, **1**, 261–265.

Meedat – Map (2008) Suivi de la population de loups en France: développer les connaissances sur la démographie, la biologie et l'écologie du loup en France. *Plan d'action national sur le loup 2008-2012, dans le contexte français d'une activité importante et traditionnelle d'élevage*. (eds Ministère de l'Ecologie, de l'Energie, du Développement durable et de l'Aménagement du territoire et Ministère de l'Agriculture et de la Pêche), pp.88, Ministère de l'Agriculture et de la Pêche, Paris.

Milner-Gulland, E.J. (1997) A stochastic dynamic programming model for the management of the saiga antelope. *Ecological Applications*, **7**, 130–142.

Moore, A.L., Hauser, C.E. & McCarthy, M.A. (2008) How we value the future affects our desire to learn. *Ecological Applications*, **18**, 1061–1069.

Murray, D.L., Smith, D.W., Bangs, E.E., Mack, C., Oakleaf, J.K., Fontaine, J. *et al.* (2010) Death from anthropogenic causes is partially compensatory in recovering wolf populations. *Biological Conservation*, **143**, 2514–2524.

Nash, S.G. & Sofer, A. (1996) *Linear and Nonlinear Programming*. McGraw-Hill, New York.

Nichols, J.D. & Williams, B.K. (2006) Monitoring for conservation. *Trends in Ecology & Evolution*, **21**, 668–673.

Nichols, J.D., Runge, M.C., Johnson, F.A. & Williams, B.K. (2007) Adaptive harvest management of North American waterfowl populations: a brief history and future prospects. *Journal of Ornithology*, **148**, S343–S349.

Nicol, S. & Chadès, I. (2011) Beyond stochastic dynamic programming: a heuristic sampling method for optimizing conservation decisions in very large state spaces. *Methods in Ecology and Evolution*, **2**, 221–228.

Norgaard, R.B. & Howarth, R.B. (1991) Sustainability and Discounting the Future. *Ecological Economics: The Science and Management of Sustainability* (ed R. Constanza), Columbia University Press, New York, New York, USA.

Pichancourt, J.B., Chadès, I., Firn, J., van Klinken, R.D. & Martin, T.G. (2012) Simple rules to contain an invasive species with a complex life cycle and high dispersal capacity. *Journal of Applied Ecology*, **49**, 52–62.

Possingham, H.P. (1997) Optimal fire management strategies for threatened species: an application of stochastic dynamic programming to state-dependent environmental decision-making. *Bulletin of the Ecological Society of America*, **78**, 813–817.

Possingham, H.P. (2001) The business of biodiversity: applying decision theory principles to nature conservation. *Tela–Environment, Economy and Society*, **9**, 44.

Punt, A. & Hilborn, R. (1997) Fisheries stock assessment and decision analysis: the Bayesian approach. *Reviews in Fish Biology and Fisheries*, **7**, 35–65.

Puterman, M.L. (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York.

Raiffa, H. & Schaifer, R. (1961) *Applied Statistical Decision Theory*. Division of Research, Harvard Business School, Boston.

Reed, W.J. (1979) Optimal escapement levels in stochastic and deterministic harvesting models. *Journal of Environmental Economics and Management*, **6**, 350–363.

Regan, H.M., Colyvan, M. & Burgman, M.A. (2002) A taxonomy and treatment of uncertainty for ecology and conservation biology. *Ecological Applications*, **12**, 618–628.

Regan, T.J., McCarthy, M.A., Baxter, P.W.J., Panetta, F.D. & Possingham, H.P. (2006) Optimal eradication: when to stop looking for an invasive plant. *Ecology Letters*, **9**, 759–766.

Richards, S.A., Possingham, H.P. & Tizard, J. (1999) Optimal fire management for maintaining community diversity. *Ecological Applications*, **9**, 880–892.

Rout, T.M., Hauser, C.E. & Possingham, H.P. (2009) Optimal adaptive management for the translocation of a threatened species. *Ecological Applications*, **19**, 515–526.

Runge, M.C. (2011) An introduction to adaptive management for threatened and endangered species. *Journal of Fish and Wildlife Management*, **2**, 220–233.

Runge, M.C. & Johnson, F.A. (2002) The importance of functional form in optimal control solutions of problems in population dynamics. *Ecology*, **83**, 1357–1371.

Schapaugh, A. W. & Tyre, A.J. (2012) A simple method for dealing with large state spaces. *Methods in Ecology and Evolution*, **3**, 949–957.

Shea, K. & Possingham, H.P. (2000) Optimal release strategies for biological control agents: an application of stochastic dynamic programming to population management. *Journal of Applied Ecology*, **37**, 77–86.

Shea, K., Thrall, P.H. & Burdon, J.J. (2000) An integrated approach to management in epidemiology and pest control. *Ecology Letters*, **3**, 150–158.

Simon, H.A. (1979) Rational Decision making in Business Organizations. *American Economic Review*, **69**, 493–513.

Spencer, P.D. (1997) Optimal harvesting of fish populations with nonlinear rates of predation and autocorrelated environmental variability. *Canadian Journal of Fisheries and Aquatic Sciences*, **54**, 59–74.

Springborn, M. & Sanchirico, J.N. (2013) *A Density Projection Approach for Non-Trivial Information Dynamics: Adaptive Management of Stochastic Natural Resources*. Working paper, University of California at Davis, Davis, California.

Stephens, P.A. & Sutherland, W.J. (1999) Consequences of the Allee effect for behaviour, ecology and conservation. *Trends in Ecology and Evolution*, **14**, 401–405.

Teeter, L., Somers, G. & Sullivan, J. (1993) Optimal forest harvest decisions: a stochastic dynamic programming approach. *Agricultural Systems*, **42**, 73–84.

Venner, S., Chadès, I., Bel-Venner, M.C., Pasquet, A., Charpillet, F. & Leborgne, R. (2006) Dynamic optimization over infinite-time horizon: web-building

strategy in an orb-weaving spider as a case study. *Journal of Theoretical Biology*, **241**, 725–733.
Walters, C.J. (1975) Optimal harvest strategies for salmon in relation to environmental variability and uncertain production parameters. *Journal of the Fisheries Board of Canada*, **32**, 1777–1784.
Walters, C.J. (1986) *Adaptive Management of Renewable Resources*. Macmillian, New York, New York, USA.
Walters, C.J. & Hilborn, R. (1976) Adaptive control of fishing systems. *Journal of the Fisheries Research Board of Canada*, **33**, 145–159.
Walters, C.J. & Hilborn, R. (1978) Ecological optimization and adaptive management. *Marine Environmental Research*, **9**, 157–188.
Wam, H.K. (2009) Economists, time to team up with the ecologists. *Ecological Economics*, **69**, 675–679.
Westphal, M.I., Pickett, M., Getz, W.M. & Possingham, H.P. (2003) The use of stochastic dynamic programming in optimal landscape reconstruction for metapopulations. *Ecological Applications*, **13**, 543–555.
Williams, B.K. (2009) Markov decision processes in natural resources management: observability and uncertainty. *Ecological Modelling*, **220**, 830–840.
Williams, B.K. (2011) Adaptive management of natural resources framework and issues. *Journal of Environmental Management*, **92**, 1346–1353.
Williams, B.K., Nichols, J.D. & Conroy, M.J. (2002) *Analysis and Management of Animal Populations*. Academic Press, San Diego, California, USA.
Wilson, K.A., McBride, M.F., Bode, M. & Possingham, H.P. (2006) Prioritizing global conservation efforts. *Nature*, **440**, 337–340.
Winkler, C. (1975) An optimization technique for the bud worm forest-pest model. International Institute for Applied Systems Analysis, Laxenburg, Austria. RM-75-l1.

*Received 28 March 2013; accepted 10 June 2013*
*Handling Editor: Robert Freckleton*

## Supporting Information

Additional Supporting Information may be found in the online version of this article.

**Appendix S1.** Deterministic dynamic programic model - value iteration over infinite horizon. Computation time ˜ 1 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz

**Appendix S2.** Deterministic dynamic programic model - policy iteration over infinite horizon. Computation time ˜ 6 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz

**Appendix S3.** (3a) Deterministic dynamic programic model - value iteration with mdptoolbox. (3b) Deterministic dynamic programic model - value iteration with mdpsolve.

**Appendix S4.** Numerical solutions representing for each population abundance the associated optimal harvest rate obtained with our R code, MDPsolve, MDPtoolbox and the analytic solution.

**Appendix S5.** Stochastic dynamic programming model with demographic stochasticity. Computation time ˜ 1 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz

```r
################################################################################
# MARESCOT ET AL.
# COMPLEX DECISIONS MADE SIMPLE: A PRIMER ON STOCHASTIC DYNAMIC PROGRAMMING
################################################################################

################################################################################
# APPENDIX 1: DETERMINISTIC DYNAMIC PROGRAMIC MODEL - VALUE ITERATION OVER INFINITE
… HORIZON
# Computation time ~ 1 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz
################################################################################

################################################################################
# STEP 1: DEFINE OBJECTIVES
################################################################################

# This is a conceptual step which does not require coding

################################################################################
# STEP 2: DEFINE STATES
################################################################################

# state space limit
K <- 250

# Vector of all possible states
states <- 0:K

################################################################################
# STEP 3: DEFINE CONTROL ACTIONS
################################################################################

# Vector of actions: rate of the population that can be removed, ranging from 0 to 1
H <- seq(0, 1, 1/(K+1))

################################################################################
# STEP 4: DEFINE DYNAMIC MODEL (WITH DEMOGRAPHIC PARAMETERS)
################################################################################

# Population growth rate
lambda <- 1.25

# Function for the exponential growth of the dynamic model
dynamic <- function(actualpop, action) {
    nextpop <- actualpop*lambda*(1-action)
    return(nextpop)
}

################################################################################
# STEP 5: DEFINE UTILITY
################################################################################

# Maximum objective threshold for population abundance
Nmax <- 250

# Minimum objective threshold for population abundance
Nmin <- 50

# Utility function
get_utility <- function(x) {
    return(ifelse(x < Nmin | x > Nmax, 0, x))
}

################################################################################
# STEP 6: SOLVE BELLMAN EQUATION WITH VALUE ITERATION
################################################################################
```

```r

# Transition matrix
transition <- array(0, dim = c(length(states), length(states), length(H)))

# Utility matrix
utility <- array(0, dim = c(length(states), length(H)))

# Fill in the transition and utility matrix
# Loop on all states
for (k in 0:K) {

    # Loop on all actions
    for (i in 1:length(H)) {

        # Calculate the transition state at the next step, given the #current state k
        # and the harvest Hi
        nextpop <- dynamic(k, H[i])

        # Compute utility
        utility[k+1,i] <- get_utility(nextpop)

        # Since the state space has a finite dimension, we cap the #next population
size
        nextpop <- min(round(nextpop), K)

        # Find next state index
        index <- which(states == nextpop)
        transition[k+1,index,i] <- 1

    } # end of action loop
} # end of state loop

# Discount factor
discount <- 0.9

# Action value vector at tmax
Vtmax <- numeric(length(states))

# Action value vector at t and t+1
Vt <- numeric(length(states))
Vtplus <- numeric(length(states))

# Optimal policy vector
D <- numeric(length(states))



# We define a factor and convergence cirterion to check for convergence #time
did_converge <- FALSE
discount <- 0.9
epsilon <-0.0001


#Tmax <- 150 defining a finite Time horizon in the case of backward #iteration

# The backward iteration consists in storing action values in the vector Vt which is
the maximum of
# utility plus the future action values for all possible next states. #Knowing the
final action
# values, we can then backwardly reset the next action value Vtplus to #the new value
Vt. We start

#for (t in (Tmax-1):1) { # the backward iteration at time T-1 since we #already defined
the action value at Tmax.
while(did_converge==FALSE) { # infinite value iteration
```

```r

    # We define a matrix Q that stores the updated action values for #all states (rows)
    # actions (columns)
    Q <- array(0, dim=c(length(states), length(H)))

    for (i in 1:length(H)) {

        # For each harvest rate we fill for all states values (row) #the ith column
(Action) of matrix Q
        # The utility of the ith action recorded for all states is #added to the
product of the transition matrix of the ith action by the #action value of all states
        Q[,i] <- utility[,i] + discount*(transition[,,i] %*% Vtplus)

    } # end of the harvest loop

    # Find the optimal action value at time t is the maximum of Q
    Vt <- apply(Q, 1, max)


    if(max(abs(Vtplus-Vt)) <= epsilon*(1-discount)/(2*discount)) did_converge <- TRUE

    # After filling vector Vt of the action values at all states, we #update the vector
Vt+1 to Vt
# and we go to the next step standing for previous time t-1, since we #iterate backward
    Vtplus <- Vt

} # end of while loop (in the case of value iteration over infinite horizon)
  #or end of the time loop (in the case of the bacward iteration)

# Find optimal action for each state
for (k in 0:K) {
    # We look for each state which column of Q corresponds to the #maximum of the last
updated value of Vt (the one at time t+1). If the #index vector is longer than 1 (if
there is more than one optimal value #we chose the minimum harvest rate)
    D[k+1] <- H[(min(which(Q[k+1,] == Vt[k+1])))]
}

################################################################################
# PLOT SOLUTION
################################################################################

plot(states, D, xlab="Population size", ylab="harvest rate")

################################################################################
# PROOF OF OPTIMALITY: COMPARE WITH ANALYTICAL SOLUTION
################################################################################

exact_policy <- rep(0,K)
for (k in 0:K) {
    exact_policy[k+1] <- max(0, 1 - K/(k*lambda))
}

# The difference between Bellman equation solution and the analytical #solution is
small:
lines(states, exact_policy)
D - exact_policy

```

```r
################################################################################
# MARESCOT ET AL.
# COMPLEX DECISIONS MADE SIMPLE: A PRIMER ON STOCHASTIC DYNAMIC PROGRAMMING
################################################################################

################################################################################
# APPENDIX 2: DETERMINISTIC DYNAMIC PROGRAMIC MODEL - POLICY ITERATION OVER INFINITE
… HORIZON
# Computation time ~ 6 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz
################################################################################

################################################################################
# STEP 1: DEFINE OBJECTIVES
################################################################################

# This is a conceptual step which does not require coding

################################################################################
# STEP 2: DEFINE STATES
################################################################################

# state space limit
K <- 250

# Vector of all possible states
states <- 0:K

################################################################################
# STEP 3: DEFINE CONTROL ACTIONS
################################################################################

# Vector of actions: rate of the population that can be removed ranging #from 0 to 1
H <- seq(0, 1, 1/(K+1))

################################################################################
# STEP 4: DEFINE DYNAMIC MODEL AND DEMOGRAPHIC PARAMETERS
################################################################################

# Population growth rate
lambda <- 1.25

# Function for the exponential growth of the dynamic model
       dynamic <- function(actualpop, action) {
    nextpop <- actualpop*lambda*(1-action)
    return(nextpop)
       }

################################################################################
# STEP 5: DEFINE UTILITY
################################################################################

# Maximum objective threshold for population abundance
Nmax <- 250

# Minimum objective threshold for population abundance
Nmin <- 50

# Utility function
get_utility <- function(x) {
    return(ifelse(x < Nmin | x > Nmax, 0, x))
}

################################################################################
# STEP 6: SOLVE BELLMAN EQUATION WITH POLICY ITERATION
################################################################################
```

```r


# Discount factor
discount <- 0.9

# Theta and delta are variables used at the end of each evaluation loop to check
# when to stop the evalution step and start the improvement step
theta <- 5.555556e-06

delta <- 0


# Boolean to check for convergence
did_converge <- FALSE

# Action value vector at t
Vt <- numeric(length(states))

# Optimal policy vector
D <- numeric(length(states))

# Vector of action values (updated at each iteration)
V <- numeric(length(states))
while(!did_converge) {

    # This control flow evaluates the policy
    repeat {

        # Loop over all states
        for (k in 0:K) {

            # We take the actual value
            v <- V[k+1]

            # Compute next population size
            # D[k+1] because arrays are indexed from 1 and not 0
            nextpop <- dynamic(k, D[k+1])

            # Compute utility
            utility <- get_utility(nextpop)

            # Truncate and cap population size
            nextpop <- min(round(nextpop), K)

            # Get index of future state
            index <- which(states==nextpop)

            # Update action value
            V[k+1] <- utility + discount*V[index]

            # Take the best improvement on the action values
            delta <- max(delta, abs(v - V[k+1]))

        } # end of the state loop

# if the best improvement on action values across the state #space is higher than a
… factor then we can test for another #action value and step to the improvement loop
        if (delta >= theta) break

    } #end of the evaluation step


    did_converge <- TRUE
    # Vector to store updated action values for each state
    Q <- numeric(length(H))
```

```
129
130     # This loop improves the policy
131     for (k in 0:K) {
132
133         # Get the action from actual policy
134         d <- D[k+1]
135
136 # Try every possible action and record their corresponding #rewards
137         for (i in 1:length(H)) {
138
139             # Compute next population size
140             nextpop2 <- dynamic(k, H[i])
141
142             # Compute utility
143             utility <- get_utility(nextpop2)
144
145             # Truncate and cap population size
146             nextpop2 <- min(round(nextpop2), K)
147
148             # Get index of future state
149             index2 <- which(states==nextpop2)
150
151             # Update Q
152             Q[i] <- utility + discount*V[index2]
153         } # End of loop on actions
154
155         # Find which action is the best
156         D[k+1] <- (min(which(Q == max(Q))) - 1)/(K+1)
157         if(D[k+1] != d) did_converge <- FALSE
158
159     } # End of improvement loop
160
161 } # End of main loop
162
163 ##############################################################################
164 # PLOT SOLUTION
165 ##############################################################################
166
167 plot(states, D, xlab="Population size", ylab="harvest rate")
168
169 ##############################################################################
170 # PROOF OF OPTIMALITY: COMPARE WITH ANALYTICAL SOLUTION
171 ##############################################################################
172
173 exact_policy <- rep(0,K)
174 for (k in 0:K) {
175     exact_policy[k+1] <- max(0, 1 - K/(k*lambda))
176 }
177
178 # The difference between Bellman equation solution and the analytical #solution is
  … small:
179 D - exact_policy
180
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MARESCOT ET AL.
% COMPLEX DECISIONS MADE SIMPLE: A PRIMER ON STOCHASTIC DYNAMIC PROGRAMMING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% APPENDIX 3a: DETERMINISTIC DYNAMIC PROGRAMIC MODEL - VALUE ITERATION WITH MDPTOOLBOX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 1: DEFINE OBJECTIVES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is a conceptual step which does not require coding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 2: DEFINE STATES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mins = 0;
maxs = 250;
ns = 251;
S = linspace(mins,maxs,ns)';
lambda = 1.25;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 3: DEFINE CONTROL ACTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mina = 0;
maxa = 250;
Umin = 50;
Umax = 250;
na = 251;
a = (linspace(mina,maxa,na)./na)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 4: DEFINE DYNAMIC MODEL AND DEMOGRAPHIC PARAMETERS
% STEP 5: DEFINE UTILITY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These two steps are done in a single loop

P=zeros(size(S,1),size(S,1), size(a,1));
R=zeros(size(S,1),size(a,1));
for i = 1:size(S,1)
    for h = 1:size(a,1)

    Snext = lambda*S(i)*(1-a(h));
    if (((Snext < Umin)|| (Snext > Umax)))
       R(i,h) = 0;
    else  R(i,h) = Snext;
    end
    Snext = min(round(Snext), maxs);
    P(i,Snext+1,h) = 1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 6: SOLVE BELLMAN EQUATION WITH VALUE ITERATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[V Policy] = mdp_value_iteration(P, R,  0.9);
plot(S, a(Policy))

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% APPENDIX 3b: DETERMINISTIC DYNAMIC PROGRAMIC MODEL - VALUE ITERATION WITH MDPSOLVE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 1: DEFINE OBJECTIVES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is a conceptual step which does not require coding

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 2: DEFINE STATES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mins = 0;
maxs = 250;
ns = 251;
S = linspace(mins,maxs,ns)';
lambda = 1.25;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 3: DEFINE CONTROL ACTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mina = 0;
maxa = 250;
Umin = 50;
Umax = 250;
na = 251;
a = (linspace(mina,maxa,na)./na)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 4: DEFINE DYNAMIC MODEL AND DEMOGRAPHIC PARAMETERS
% STEP 5: DEFINE UTILITY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate all possible State-action combinations
X = rectgrid(S,a);
Ix=getI(X,1);
nx=size(X,1);
% Calculate futur states after applying each action to each current state
Snext = lambda.*X(:,1).*(1-X(:,2));

% Fill the reward vector
reward = (Snext(:)>=Umin & Snext(:) <=Umax);
R = zeros(size(Snext,1),1);
R(reward,:) = Snext(reward);

% Build the transition matrix
Snext = min(round(Snext),maxs);
g = @(X) (lambda.*X(:,1).*(1-X(:,2)));
goptions = struct('cleanup', 0, 'recinterp', 0);
P = g2P(g, S, X, goptions);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 6: SOLVE BELLMAN EQUATION WITH VALUE ITERATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
model.Ix=Ix;
model.X = X;
model.P = P;
delta = 0.9;
model.d = delta;
model.R = R;
result = mdpsolve(model);
```

```
131 plot(result.Xopt(:,1),result.Xopt(:,2))
132
```

Appendix S4: Numerical solutions representing for each population abundance the associated optimal harvest rate obtained with our R code, MDPsolve, MDPtoolbox and the analytic solution.

| Abundance | R value iteration | R policy iteration | MDPtoolbox | MDPsolve | Analytic solution |
|---|---|---|---|---|---|
| from 0 to 200 | 0 | 0 | 0 | 0 | 0 |
| 201 | 0,008 | 0,008 | 0,008 | 0,008 | 0,005 |
| 202 | 0,012 | 0,012 | 0,012 | 0,012 | 0,010 |
| 203 | 0,016 | 0,016 | 0,016 | 0,016 | 0,015 |
| 204 | 0,020 | 0,020 | 0,020 | 0,020 | 0,020 |
| 205 | 0,028 | 0,028 | 0,028 | 0,028 | 0,024 |
| 206 | 0,032 | 0,032 | 0,032 | 0,032 | 0,029 |
| 207 | 0,036 | 0,036 | 0,036 | 0,036 | 0,034 |
| 208 | 0,040 | 0,040 | 0,040 | 0,040 | 0,038 |
| 209 | 0,044 | 0,044 | 0,044 | 0,044 | 0,043 |
| 210 | 0,048 | 0,048 | 0,048 | 0,048 | 0,048 |
| 211 | 0,056 | 0,056 | 0,056 | 0,056 | 0,052 |
| 212 | 0,060 | 0,060 | 0,060 | 0,060 | 0,057 |
| 213 | 0,064 | 0,064 | 0,064 | 0,064 | 0,061 |
| 214 | 0,068 | 0,068 | 0,068 | 0,068 | 0,065 |
| 215 | 0,072 | 0,072 | 0,072 | 0,072 | 0,070 |
| 216 | 0,076 | 0,076 | 0,076 | 0,076 | 0,074 |
| 217 | 0,080 | 0,080 | 0,080 | 0,080 | 0,078 |
| 218 | 0,084 | 0,084 | 0,084 | 0,084 | 0,083 |
| 219 | 0,088 | 0,088 | 0,088 | 0,088 | 0,087 |
| 220 | 0,092 | 0,092 | 0,092 | 0,092 | 0,091 |
| 221 | 0,096 | 0,096 | 0,096 | 0,096 | 0,095 |
| 222 | 0,100 | 0,100 | 0,100 | 0,100 | 0,099 |
| 223 | 0,104 | 0,104 | 0,104 | 0,104 | 0,103 |
| 224 | 0,108 | 0,108 | 0,108 | 0,108 | 0,107 |
| 225 | 0,112 | 0,112 | 0,112 | 0,112 | 0,111 |
| 226 | 0,116 | 0,116 | 0,116 | 0,116 | 0,115 |
| 227 | 0,120 | 0,120 | 0,120 | 0,120 | 0,119 |
| 228 | 0,124 | 0,124 | 0,124 | 0,124 | 0,123 |
| 229 | 0,127 | 0,127 | 0,127 | 0,128 | 0,127 |
| 230 | 0,131 | 0,131 | 0,131 | 0,132 | 0,130 |
| 231 | 0,135 | 0,135 | 0,135 | 0,136 | 0,134 |
| 232 | 0,139 | 0,139 | 0,139 | 0,139 | 0,138 |
| 233 | 0,143 | 0,143 | 0,143 | 0,143 | 0,142 |
| 234 | 0,147 | 0,147 | 0,147 | 0,147 | 0,145 |
| 235 | 0,151 | 0,151 | 0,151 | 0,151 | 0,149 |
| 236 | 0,155 | 0,155 | 0,155 | 0,155 | 0,153 |
| 237 | 0,159 | 0,159 | 0,159 | 0,159 | 0,156 |
| 238 | 0,163 | 0,163 | 0,163 | 0,163 | 0,160 |
| 239 | 0,163 | 0,163 | 0,163 | 0,163 | 0,163 |
| 240 | 0,167 | 0,167 | 0,167 | 0,167 | 0,167 |

| | | | | | |
|---|---|---|---|---|---|
| 241 | 0,171 | 0,171 | 0,171 | 0,171 | 0,170 |
| 242 | 0,175 | 0,175 | 0,175 | 0,175 | 0,174 |
| 243 | 0,179 | 0,179 | 0,179 | 0,179 | 0,177 |
| 244 | 0,183 | 0,183 | 0,183 | 0,183 | 0,180 |
| 245 | 0,187 | 0,187 | 0,187 | 0,187 | 0,184 |
| 246 | 0,187 | 0,187 | 0,187 | 0,187 | 0,187 |
| 247 | 0,191 | 0,191 | 0,191 | 0,191 | 0,190 |
| 248 | 0,195 | 0,195 | 0,195 | 0,195 | 0,194 |
| 249 | 0,199 | 0,199 | 0,199 | 0,199 | 0,197 |
| 250 | 0,203 | 0,203 | 0,203 | 0,203 | 0,200 |

```r
################################################################################
# MARESCOT ET AL.
# COMPLEX DECISIONS MADE SIMPLE: A PRIMER ON STOCHASTIC DYNAMIC PROGRAMMING
################################################################################

################################################################################
# APPENDIX 5: STOCHASTIC DYNAMIC PROGRAMMING MODEL WITH DEMOGRAPHIC STOCHASTICITY
# Computation time ~ 1 min. on an Intel Xeon CPU X5670 Westmere @2.93 GHz
################################################################################

################################################################################
# STEP 1: DEFINE OBJECTIVES
################################################################################

# This is a conceptual step which does not require coding

################################################################################
# STEP 2: DEFINE STATES
################################################################################

# state space limit
K <- 250

# Vector of all possible states
states <- 0:K

################################################################################
# STEP 3: DEFINE CONTROL ACTIONS
################################################################################

# Vector of actions: rate of the population that can be removed, ranging #from 0 to 1
H <- seq(0, 1, 1/(K+1))

################################################################################
# STEP 4: DEFINE DYNAMIC MODEL (WITH DEMOGRAPHIC PARAMETERS)
################################################################################

# Population growth rate
lambda <- 1.25

# Function for the exponential growth of the dynamic model
dynamic <- function(actualpop, action) {
    nextpop <- actualpop*lambda*(1-action)
    return(nextpop)
}

################################################################################
# STEP 5: DEFINE UTILITY
################################################################################

# Maximum objective threshold for population abundance
Nmax <- 250

# Minimum objective threshold for population abundance
Nmin <- 50

# Utility function
get_utility <- function(x) {
    return(ifelse(x < Nmin | x > Nmax, 0, x))
}

################################################################################
# STEP 6: SOLVE BELLMAN EQUATION WITH VALUE ITERATION
################################################################################

```

```r
# Transition matrix
transition <- array(0, dim = c(length(states), length(states), length(H)))

# Utility matrix
utility <- array(0, dim = c(length(states), length(H)))

# Fill in the transition and utility matrix
# Loop on all states
for (k in 0:K) {

    # Loop on all actions
    for (i in 1:length(H)) {

# Calculate the transition state at the next step, given the #current state k and the harvest Hi
        nextpop <- dynamic(k, H[i])

    # Implement demographic stochasticity by drawing
      #probability from a Poisson density function
        transition[k+1,,i] <- dpois(states,nextpop)
        # We need to correct this density for the final capping state
        transition[k+1,K+1,i] <- 1 - sum(transition[k+1,-(K+1),i])

        # Compute utility
        utility[k+1,i] <- get_utility(nextpop)

    } # end of action loop
} # end of state loop

# Discount factor
discount <- 0.9

# Action value vector at tmax
Vtmax <- numeric(length(states))

# Action value vector at t and t+1
Vt <- numeric(length(states))
Vtplus <- numeric(length(states))

# Optimal policy vector
D <- numeric(length(states))

# Time horizon
Tmax <- 150

# The backward iteration consists in storing action values in the vector Vt which is the maximum of
# utility plus the future action values for all possible next states. Knowing the final action
# values, we can then backwardly reset the next action value Vtplus to the new value Vt. We start
# The backward iteration at time T-1 since we already defined the action #value at Tmax.
for (t in (Tmax-1):1) {

# We define a matrix Q that stores the updated action values for #all states (rows)
    # actions (columns)
    Q <- array(0, dim=c(length(states), length(H)))

    for (i in 1:length(H)) {

# For each harvest rate we fill for all states values (row) #the ith column (Action) of matrix Q
# The utility of the ith action recorded for all states is #added to the product of the transition matrix of the ith #action by the action value of all states
```

```
        Q[,i] <- utility[,i] + discount*(transition[,,i] %*% Vtplus)

    } # end of the harvest loop

    # Find the optimal action value at time t is the maximum of Q
    Vt <- apply(Q, 1, max)

# After filling vector Vt of the action values at all states, we #update the vector
Vt+1 to Vt and we go to the next step standing #for previous time t-1, since we iterate
backward
    Vtplus <- Vt

} # end of the time loop

# Find optimal action for each state
for (k in 0:K) {
# We look for each state which column of Q corresponds to the #maximum of the last
updated value
# of Vt (the one at time t+1). If the index vector is longer than 1 #(if there is more
than one optimal value we chose the minimum #harvest rate)
    D[k+1] <- H[(min(which(Q[k+1,] == Vt[k+1])))]
}

#############################################################################
# PLOT SOLUTION
#############################################################################

plot(states, D, xlab="Population size", ylab="harvest rate")

#############################################################################
# PROOF OF OPTIMALITY: COMPARE WITH ANALYTICAL SOLUTION
#############################################################################

exact_policy <- rep(0,K)
for (k in 0:K) {
    exact_policy[k+1] <- max(0, 1 - K/(k*lambda))
}

# The difference between Bellman equation solution and the analytical #solution is
small:
lines(states, exact_policy)
D - exact_policy
```