Methods in Ecology and Evolution

British Ecological Society

Methods in Ecology and Evolution 2012, **3**, 949–957 doi: 10.1111/j.2041-210X.2012.00242.x

A simple method for dealing with large state spaces

Adam W. Schapaugh* and Andrew J. Tyre

School of Natural Resources, University of Nebraska-Lincoln, Hardin Hall, 3310 Holdrege Street, Lincoln, NE, 68510, USA

Summary

1. Most sequential decision-making problems in conservation can be viewed conceptually and modelled as a Markov decision process. The goal in this context is to construct a *policy* that associates each state of the system with a particular action. This policy should offer optimal performance in the sense of maximizing or minimizing a specified conservation objective

2. Dynamic programming algorithms rely on explicit enumeration to derive the optimal policy. This is problematic from a computational perspective as the size of the state space grows exponentially with the number of state variables.

3. We present a state aggregation method where the idea is to capture the most important aspects of the original Markov decision process, find an optimal policy over this reduced space and use this as an approximate solution to the original problem.

4. Applying the aggregation method to a species reintroduction problem, we demonstrate how we were able to reduce the number of states by 75% and reduce the size of the transition matrices by almost 94% (324 vs. 5184), and the abstract action matched the optimal action more than 86% of the time.

5. We conclude that the aggregation method is not a panacea for the curse of dimensionality, but it does advance our ability to construct approximately optimal policies in systems with large state spaces.

Key-words: abstraction, curse of dimensionality, Markov decision process, state space, stochastic dynamic programming

Introduction

Markov decision processes (MDPs) have come to play an increasingly important role in conservation planning research, forming the basic model for recent investigations into metapopulation management (e.g. Westphal *et al.* 2003), invasive species control (e.g. Bogich & Shea 2008), translocation (e.g. Tenhumberg *et al.* 2004) and sequential reserve selection (e.g. Costello & Polasky 2004). The goal in such applications is to construct a *policy* that associates each state of the system with a particular action. This policy should offer optimal performance in the sense of maximizing or minimizing a specified conservation objective (Possingham *et al.* 2001).

A standard technique for solving finite-horizon MDPs is backward induction. This dynamic programming algorithm relies on an extensional representation of the state space and explicit enumeration (i.e. every state is visited at every time step) to derive the optimal policy. Specifying the effects of actions in terms of state transitions can be problematic, however, because the size of the state space grows exponentially with the number of state variables. For example, a problem with 15 binary (0 or 1) state variables (2^{15} = 32 768 states) would require transition matrices with 1 073 741 824 entries to represent the effects of each action. This 'curse of dimensionality' (Bellman 1961) has an impact on the feasibility of the specification and solution of MDPs in the context of real-world conservation planning.

A great deal of emphasis in the artificial intelligence community has been placed on dodging the curse of dimensionality by means of approximation (for discussion, see Boutilier, Dean & Hanks 1999; Li, Walsh & Littman 2006; Chapter 6 of Bertsekas 2007; Powell 2007). One class of methods involves restricting search to locally accessible regions of the state space (for examples in conservation, see Nicol *et al.* 2010; Nicol & Chades 2011). These methods generally take advantage of the fact that the current state of the system may be known, forming the basis of what we view as an abbreviated version of decision tree search. Each action at the current state forms the first level of the tree. A generative model is then used to simulate the possible future states given each action, which are placed at the second level of the tree. The third level has the actions applicable at the states at the second level and so on, looking ahead a defined number of steps. A sub-MDP calculation is then carried out on the simulated future states, which approximates the optimal action for the root of the search tree (i.e. the current state). The important point to observe is that attention is restricted to the locally accessible regions of the state space. This can have advantages over conventional dynamic programming techniques, especially if only a fraction of the state

*Correspondence author. Email: adam.schapaugh@huskers.unl.edu

space is connected to the current state in a given number of look-ahead steps (Boutilier, Dean & Hanks 1999).

The primary disadvantage of these local search methods is that many states are ignored in policy construction. The work of Nicol *et al.* (2010) and Nicol & Chades (2011) can be viewed as applications of 'online' methods that handle the problem in a serial fashion. They sacrifice the optimal policy for a fast, local approximation that applies only to the current state. Here, we explore a way that also sacrifices the optimal policy but does so for an approximation that applies to every state in the state space. We describe an abstraction method proposed by Dearden & Boutilier (1997). The method is a form of state aggregation; more specifically, we use the reward structure of the problem to select a subset of the state variables whose impact on the value of a state is minimal, negligible or absent. These state variables are then deleted from the problem description. The abstract state space (which is exponentially smaller than the original) is found by aggregating all states that agree on the values of the state variables that remain. The idea is to capture the most important aspects of the original MDP, find an optimal policy over this reduced space and use this as an approximate solution to the original problem. We first review MDPs and the backward induction algorithm, then the method, problems with it and interesting conclusions are illustrated with two problems in conservation planning.

Markov Decision Processes

Markov decision processes (Bellman 1957) provide a mathematical framework for modelling sequential decision-making problems. As the name implies, MDPs are an extension of Markov chains; the difference is the addition of actions (to influence the state of the system) and rewards (giving motivation). An MDP is defined by the following components: a set of states: $s \in S$; a set of actions: $x \in X$; a state transition function: T; and a reward: $R(s, x)$ for executing action x in state s. At each stage (or time step), the decision-maker observes the state of the system and selects an action. The state and action choice produce two results: the decision-maker receives a reward and the system transitions from one stage to the next. These transitions are not deterministic; instead, each action is represented by a transition matrix of size $S \times S$, containing the probability that performing x in state s will move the system to state s', that is, $\Pr(S_{t+1} = s' | S_t = s,\ x_t = x)$ (Putterman 1994).

The problem facing the decision-maker can be viewed as deciding which action to perform given the current state of the system. More generally, we seek a policy, π, which is defined as a mapping from the state and stage to actions, that is, π: $s \times t \rightarrow x$. When a problem's horizon is finite (i.e. a fixed number of stages, T), a standard technique for optimal policy construction is backward induction. The algorithm proceeds as follows:

1 Set $t = T + 1$ and $V^{T+1}(s) = f(s)$ for all $s \in S_{T+1}$ (terminal value is a function of the state)

2 Substitute $t - 1$ for t and compute $V(t, T, s)$ for each $s \in S$ using:

$$V(t, T, s) = \max_{x \in X} \left\{ R(s, x) + \sum_{s'} \Pr(s' | S, x) V(t+1, T, s') \right\}$$

3 If $t = 1$, stop, otherwise return to step (2).

In words, the algorithm chooses maximizing actions in reverse order. At the terminal time, T, the best action in each state is selected. In $T - 1$, $V(t, T, s)$ is found by selecting the action that maximizes the immediate reward plus the expected terminal reward. These expected values are calculated by weighting all possible outcomes over the next time step by their probability of occurrence and summing the results. This process is repeated in stage $T - 2$, $T - 3$ and so on, until stage $t = 1$. This procedure accomplishes one primary objective: it finds the set of actions that maximize the Bellman equation in Step (2). This set of actions is the optimal policy. For more discussion on MDPs and dynamic programming techniques, see Putterman (1994) and Clark & Mangel (2000).

Materials and methods

Backward induction and other dynamic programming algorithms use an extensional representation for the set of states. We mentioned previously how the transition function for this representation requires a set of $S \times S$ matrices, one matrix for each action. For problems with a large number of states, the specification and storage requirements for these action descriptions can be cumbersome, if not prohibitive. Dearden & Boutilier (1997) provide an appealing approximation technique to overcome this difficulty. For clarity and ease of presentation, we describe the algorithm in four steps: (1) decide which state variables are most important (this defines an abstract state space $\tilde{S}$); (2) for each action, build an abstract transition function $\tilde{T}$; (3) simplify action descriptions $\tilde{A}$; and (4) construct the abstract reward function $\tilde{R}$. Once we have constructed the abstract MDP $(\tilde{S}, \tilde{A}, \tilde{T}, \tilde{R})$, we can compute the optimal abstract policy, $\tilde{\pi}$, using any standard solution technique.

ABSTRACT STATE SPACE

Constructing an abstract MDP requires that we identify the state variables that must be retained in the problem. We first identify a set of immediately relevant (IR) state variables. This set is formed by examining the reward structure of the problem and selecting the state variables that have the greatest impact on the reward for each state. The larger this set is, the more accurate the abstraction will be. Thus, by varying the size of the IR set, we can examine the balance between the quality of the abstraction and the feasibility of the specification. Specifically, we examine each state variable that appears in the reward function and calculate the maximum range of the reward function for each of its values. In general, state variables with smaller ranges have greater overall effect on reward than variables with larger ranges; these should be retained first.

The IR set contains state variables that appear in the reward function. It does not, however, include every state variable needed for the abstraction. For example, in a problem where the reward is large if variable X is true and small otherwise, the IR set would be $\{X\}$. But if an action that makes X true requires a second state variable, say Y, to be true also, then Y is clearly relevant: failing to retain Y may not give the decision-maker the ability to affect X as they should. To define the set of relevant (R) state variables, Dearden & Boutilier (1997) adopt a

probabilistic analog of STRIPS (Fikes & Nilsson 1971). In the STRIPS representation, actions are represented using lists of *effects* or sets of state variables that change value when an action is executed. The state that results when an action is executed at state s is simply the result of applying effect E to s, that is $E(s) = (s\backslash\{P{:}P \notin E\} \cup E)$. Following Kushmerick, Hanks & Weld (1994), we assume that the conditions where an action can have different effects are described by a set of *discriminants*, which are a set of mutually exclusive and exhaustive formulae that partition the state space. We denote $SV(d^i)$ to be the set of state variables occurring in d^i, where d^i is the unique action discriminant such that s is contained in d^i. Because we are dealing with stochastic actions, a number of effects might occur with nonzero probability. Specifically, for each d^i, we assign a *stochastic effects list* of the form $(E^i_1, P^i_1, \ldots, E^i_n, P^i_n)$, where each E^i_j is an effect and each P^i_j is the probability that the effect will occur. An action now induces a probability distribution over the possible resulting states.

The set of *relevant* (R) state variables is defined as the smallest set satisfying the two following conditions: (1) IR ⊆ R; (2) if $q \in$ R and for an effect E^i_j, $q \in SV(E^i_j)$, then $SV(d^i) \subseteq$ R. What the second condition is saying is that only the state variables in a discriminant that might probabilistically lead to a relevant effect are deemed important. To generate the set of relevant state variables, we simply back-step through action descriptions to see what state variables influence those in the IR set, what state variables influence those and so on (for an automated approach, see figure 6, Dearden & Boutilier 1997]). After finding the set of relevant state variables, the abstract state space is found by aggregating all of the states in the original MDP that agree on the values of the state variables in the relevant set. By treating each aggregate cluster as a state in the abstract MDP, we ignore the details of the state variables that do not appear in R.

ABSTRACT TRANSITION FUNCTION

In addition to the abstract state space, we need a state transition function that is compatible with the aggregate clusters. The definition of R (in particular, the requirement that all state variables of a discriminant be added to R whenever the corresponding effect is in R) ensures that the states in any given cluster have the same transition probabilities for each action. More accurately, every state in a cluster has the same probability of transitioning to another cluster. As a result, we can assign the transition probability for any state in the cluster to the cluster itself.

ABSTRACT ACTIONS

The fact that we can assign the transition probability for any state in the cluster to the cluster itself permits a simple syntactic procedure to build abstract action descriptions; we simply delete all reference to irrelevant state variables from the actions in the original problem. The steps needed to construct an abstract action description, given the set R: (1) delete irrelevant state variables from each d^i and E^i_j; and (2) for each d^i, collapse any effects that have become identical into a single reduced effect with probability $\sum P^i_j$.

ABSTRACT REWARD FUNCTION

The abstract reward function must associate a reward with each cluster. We assign the midpoint of the range of rewards for the states in $\tilde{s}$. More formally, let $\min(\tilde{s})$ and $\max(\tilde{s})$ denote the minimum and maximum values of the set $R(s) : s \in \tilde{s}$,respectively. The abstract reward function is

$$\tilde{R}(\tilde{s}) = \frac{\min(\tilde{s}) + \max(\tilde{s})}{2}.$$

This choice of $\tilde{R}$ minimizes the maximum difference between $\tilde{R}(\tilde{s})$ and R(s).

BOUNDING THE ERROR OF AN ABSTRACT MDP

We can place a bound on solution quality by calculating the value lost by using the abstract policy in the original problem. Quality in this sense is characterized by the maximum reward span of the abstract MDP, where we first define the reward span of a cluster as the maximum range of possible rewards for that cluster:

$$\text{span}(\tilde{s}) = \max(\tilde{s}) - \min(\tilde{s}),$$

where $\min(\tilde{s})$ and $\max(\tilde{s})$are defined as mentioned previously. Thus, the reward span for a cluster is twice the maximum degree to which the estimate $\tilde{R}(\tilde{s})$ of the reward associated with a state differs from the true reward $R(s)$ for that state. Then, the maximum reward span, δ, over all the clusters in the abstract state space $\tilde{S}$ is

$$\delta = \max_{\tilde{s} \in \tilde{S}} \{\text{span}(\tilde{S})\}.$$

Perhaps, the most important result presented by Dearden & Boutilier (1997) is that by utilizing an abstract solution in the original problem, a decision-maker is guaranteed to lose no more than a reward of δ per stage of the process. In addition, the smaller the reward span of the clusters used in the abstract MDP, the better the performance guarantees on the abstract solution.

EXAMPLES

We use two examples to illustrate the abstraction algorithm. The first example has a trivial number of states, but its small size lends itself to an easy description of how the abstraction algorithm works. We develop this example step by step, including the STRIPS-style action and reward representations, and we compare the performance of the abstract solution with the true optimal policy. We then present a much larger example, one in which problem size prohibits the use of standard dynamic programming techniques.

Species reintroduction

In our first example, we suppose that an agency is charged with the task of reintroducing a species to a portion of its former range. We assume that individuals may be captured from an existing source population, transferred and held at a captive facility, then released into the new target population. The agency is rewarded for establishing the target population, but they are penalized if the existing source population is eradicated (because of capture and translocation). Additionally, there is concern about disease transmission while wild individuals are held at the captive facility. The agency is again penalized if this transmission occurs. This sequential decision-making problem is characterized by five state variables (variable and action names shown in *italics*; values each variable can take are shown in parentheses): *Source Population Size* (0; 1–20; 21+); *Captive Population Size* (0; 1–20); *Target Population Size* (0; 1–20; 21+); *Captive Infected* (True; False); *Wild Infected* (True; False). The agency has four actions at their disposal, three of which may fail: *Capture* (i.e. remove individuals from *Source Population*); *Release* (i.e. release individuals into *Target Population*); *Isolate* (i.e. quarantine wild individuals from captive population); and *Do Nothing*

Table 1. Stochastic STRIPS-style action representation for the species reintroduction problem

Action	Discriminant	Effect	*P*
Capture	*Source Population Size* (1–20); *Captive Population Size* (0)	*Source Population Size* (0); *Captive Population Size* (0)	0·01
		Source Population Size (0); *Captive Population Size* (1–20)	0·09
		Source Population Size (1–20); *Captive Population Size* (0)	0·09
		Source Population Size (1–20); *Captive Population Size* (1–20)	0·81
	Source Population Size (21+); *Captive Population Size* (0)	*Source Population Size* (1–20); *Captive Population Size* (0)	0·01
		Source Population Size (1–20); *Captive Population Size* (1–20)	0·09
		Source Population Size (21+); *Captive Population Size* (0)	0·09
		Source Population Size (21+); *Captive Population Size* (1–20)	0·81
	Source Population Size (1–20); *Captive Population Size* (1–20)	*Source Population Size* (0); *Captive Population Size* (1–20)	0·10
		Source Population Size (1–20); *Captive Population Size* (1–20)	0·90
	Source Population Size (21+); *Captive Population Size* (1–20)	*Source Population Size* (1–20); *Captive Population Size* (1–20)	0·10
		Source Population Size (21+); *Captive Population Size* (1–20)	0·90
Isolate	*Captive Infected* (True); *Wild Infected* (False)	*Captive Infected* (True); *Wild Infected* (False)	0·90
		Captive Infected (True); *Wild Infected* (True)	0·10
Release	*Captive Population Size* (1–20); *Target Population Size* (0)	*Captive Population Size* (0); *Target Population Size* (0)	0·10
		Captive Population Size (0; *Target Population Size* (1–20)	0·90
	Captive Population Size (1–20); *Target Population Size* (1–20)	*Captive Population Size* (0); *Target Population Size* (1–20)	0·10
		Captive Population Size (0); *Target Population Size* (21+)	0·90
	Captive Population Size (1–20); *Target Population Size* (21+)	*Captive Population Size* (0); *Target Population Size* (21+)	1·00

(i.e. just as it sounds). The effects of these actions and their probabilities are shown in Table 1.

There are three state variables that influence the reward assigned to a state: *Source Population Size*; *Target Population Size*; and *Wild Infected* [Table 2 (Nominal Reward)]. The influence of the first two state variables is relatively large, while that of *Wild Infected* is less substantial. *Source Population Size* has a range of 50 when equal to {0}, a range of 50 when equal to {1–20} and a range of 50 when equal {21+}. *Target Population Size* has a range of 60 when equal to {0}, a range of 50 when equal to {1–20} and a range of 60 when equal {21+}. Both are better candidates for inclusion in the IR set than *Wild Infected*, which has a range of 90 whether {True} or {False} (Table 2). We thus set IR = {*Source Population Size*, *Target Population Size*}. To construct R, we examine the discriminants of the actions that affect these two state variables (Table 1). In doing so, we notice that *Source Population Size* influences *Captive Population Size* through action *Capture*, which, in turn, influences *Target Population Size* through action *Release*. We end up with R = {*Source Population Size*, *Captive Population Size, and Target Population Size*}. The abstract state space consists of those subsets of eighteen states that agree on the assignment to these three state variables, but disagree on the values of the discarded state variables *Captive Infected* and *Wild Infected*. Notice that $|\tilde{S}| = 18$, while $|S| = 72$.

Landscape restoration

Our second example involves deciding how to distribute limited resources between sites for restoration. We assume that an agency is allocated a fixed annual budget which is used to fund restoration projects. The agency is rewarded for sites that are in restored condition

Table 2. STRIPS-style reward representation for the species reintroduction problem

Discriminant	(Nominal) Reward	(Uniform) Reward	(Preferred) Reward
Source Population Size (0); *Target Population Size* (0); *Wild Infected* (True)	0	0	0
Source Population Size (0); *Target Population Size* (0); *Wild Infected* (False)	10	25	40
Source Population Size (0); *Target Population Size* (1–20); *Wild Infected* (True)	25	20	10
Source Population Size (0); *Target Population Size* (1–20); *Wild Infected Wild Infected* (False)	35	45	50
Source Population Size (0); *Target Population Size* (21+); *Wild Infected* (True)	40	35	25
Source Population Size (0); *Target Population Size* (21+); *Wild Infected* (False)	50	60	65
Source Population Size (1–20); *Target Population Size* (0); *Wild Infected* (True)	35	25	20
Source Population Size (1–20); *Target Population Size* (0); *Wild Infected* (False)	45	50	60
Source Population Size (1–20); *Target Population Size* (1–20); *Wild Infected* (True)	60	45	30
Source Population Size (1–20); *Target Population Size* (1–20); *Wild Infected* (False)	70	70	70
Source Population Size (1–20); *Target Population Size* (21+); *Wild Infected* (True)	75	60	45
Source Population Size (1–20); *Target Population Size* (21+); *Wild Infected* (False)	85	85	85
Source Population Size (21+); *Target Population Size* (0); *Wild Infected* (True)	50	40	35
Source Population Size (21+); *Target Population Size* (0); *Wild Infected* (False)	60	65	75
Source Population Size (21+); *Target Population Size* (1–20); *Wild Infected* (True)	65	60	45
Source Population Size (21+); *Target Population Size* (1–20); *Wild Infected* (False)	75	85	85
Source Population Size (21+); *Target Population Size* (21+); *Wild Infected* (True)	90	75	60
Source Population Size (21+); *Target Population Size* (21+); *Wild Infected* (False)	100	100	100

at the end of the planning horizon. The objective is to maximize the number of sites in restored condition while accounting for the fact that different sites contribute differently to the reward assigned to a state, unmanaged sites may degrade over time, and many restoration projects fail to meet their goals. The likelihood of success declines as more sites are managed (because resources are spread more thinly). The agency has four actions at their disposal: *Top Two* (i.e. manage the two most valuable sites); *Top Five* (i.e. manage the five most valuable sites); *Top Ten* (i.e. manage the ten most valuable sites); and *All* (i.e. manage all sites).

We consider a landscape with 20 sites, where each site is a state variable and takes on a value of 0 or 1 (representing degraded or restored condition, respectively). Every state variable influences the reward assigned to a state, and reflecting differences in habitat quality, quantity, etc., sites contribute differently to the reward assigned to a state. Sites 1–5 have a value of 1 when restored and a range of 214 whether {0} or {1}, Sites 6–10 have a value of 5 when restored and a range of 210, Sites 11–15 have a value of 15 when restored and a range of 200, Sites 16–18 have a value of 20 when restored and a range of 195, and Sites 19–20 have a value of 25 when restored and a range of 190. We thus set IR =R = Sites 11–20 (IR = R because every state variable influences the reward assigned to a state). Notice that $|\tilde{S}| = 1024$, while $|S| = 1\ 048\ 576$. We used Program R v 2·10·0 (R Development Core Team 2009) for all analyses.

Results

FEASIBILITY OF THE SPECIFICATION

The utility of a particular abstraction is a function of the feasibility of the specification. In our reintroduction problem, the number of states in the original MDP was 72, which required four transition matrices with 5184 entries. The number of states in the abstract MDP was 18, which required three transition matrices with 324 entries. In the landscape restoration problem, the number of states in the original MDP was 1 048 576, which would require four transition matrices with 1 099 511 627 776 entries. The number of states in the abstract MDP was 1024, which required three transition matrices with 1 048 576 entries.

QUALITY OF THE APPROXIMATION

The utility of an abstraction is also a function of the quality of the approximation. Recall that we are trying to capture the most important aspects of the original MDP, find an optimal policy over this reduced space and use this as an approximate solution to the original problem. The extent to which the abstract policy agrees with the optimal policy provides a measure of the quality of the approximation. Accordingly, we can measure quality in three ways. First, we can think of the number of errors as the number of states at which the abstract action differs from the optimal action. Second, we can calculate the value lost by using the abstract policy as a solution in the original problem. Here, loss occurs when a suboptimal decision is carried out in the original state space. Third, we can compare the values for the abstract and optimal policies, describing the deviation between the two as another form of error.

SPECIES REINTRODUCTION

The *a priori* error bound on the value lost per stage was 10 (or 10% of the possible range of optimal values). This is a worse-case bound, and we found that the actual maximum value lost was 5·90. Using the abstract policy as an approximate solution to the original problem, the *average* value lost per stage was 0·61, or 2·33%. The average error in the value of a state was 4·84 (SE = 1·22), with a maximum error of 9·50. The number of optimal actions chosen by the abstract policy per stage was 62 (of 72) or more than 86% (Fig. 1). Most importantly, the abstract policy was *never* wrong when the optimal action involved *Capture* or *Release*. As we would expect given the method of construction, the disagreement between the abstract and optimal policies occurred when *Captive Infected* was True and *Wild Infected* was False, and there were only two feasible actions: *Isolate* and *Do Nothing*. In these states, the optimal action in the original MDP was to *Isolate*; in the abstract MDP, the *Isolate* action was removed and decision-maker was forced to *Do Nothing*.

We suspected that the success (or quality) of this abstraction was due, at least in part, to the way we assigned reward to each state (Table 2). Specifically, we had defined sub-goals (*Source Population Size*, *Target Population Size*) whose contributions to the value function were larger than that of the other subgoal (*Wild Infected*). To push the abstraction, or find conditions where the quality of the approximation deteriorated, we tried two alterative reward definitions. In the first alternative (Uniform Reward, Table 2), we eliminated the preference for the first two sub-goals, making *Wild Infected* (nearly) as important as the other two. In the second alternative (Preferred Reward; Table 2), we valued *Wild Infected* more than either of the other two subgoals. A comparison of these two alternatives against the 'nominal' reward assignment is shown in Table 3. Importantly, the quality of the abstract policy dropped as the discrepancy in preference for *Source Population Size*, *Target Population Size* vs. *Wild Infected* was reduced, and then eliminated. The *a priori* error bound and actual maximum value lost grew to 25 and 14·76, respectively, for the Uniform Reward alternative, and the number of errors in action choice grew to 16. Under the Preferred Reward alternative, the *a priori* and actual maximum value lost grew to 40 and 23·62, respectively, and the number of errors in action choice was 20 (Table 3).

LANDSCAPE RESTORATION

In this much larger problem, the *a priori* error bound on the value lost per stage was 30 (or 13·9% of the possible range of optimal values). As a point of comparison, we calculated the same error bound for other possible abstractions (retaining different numbers of state variables), which is shown in Table 4. We found that as more state variables were included in the abstraction, the better the performance guarantees on the resulting policy. This result illustrates that solution quality is a function of reward span not of problem

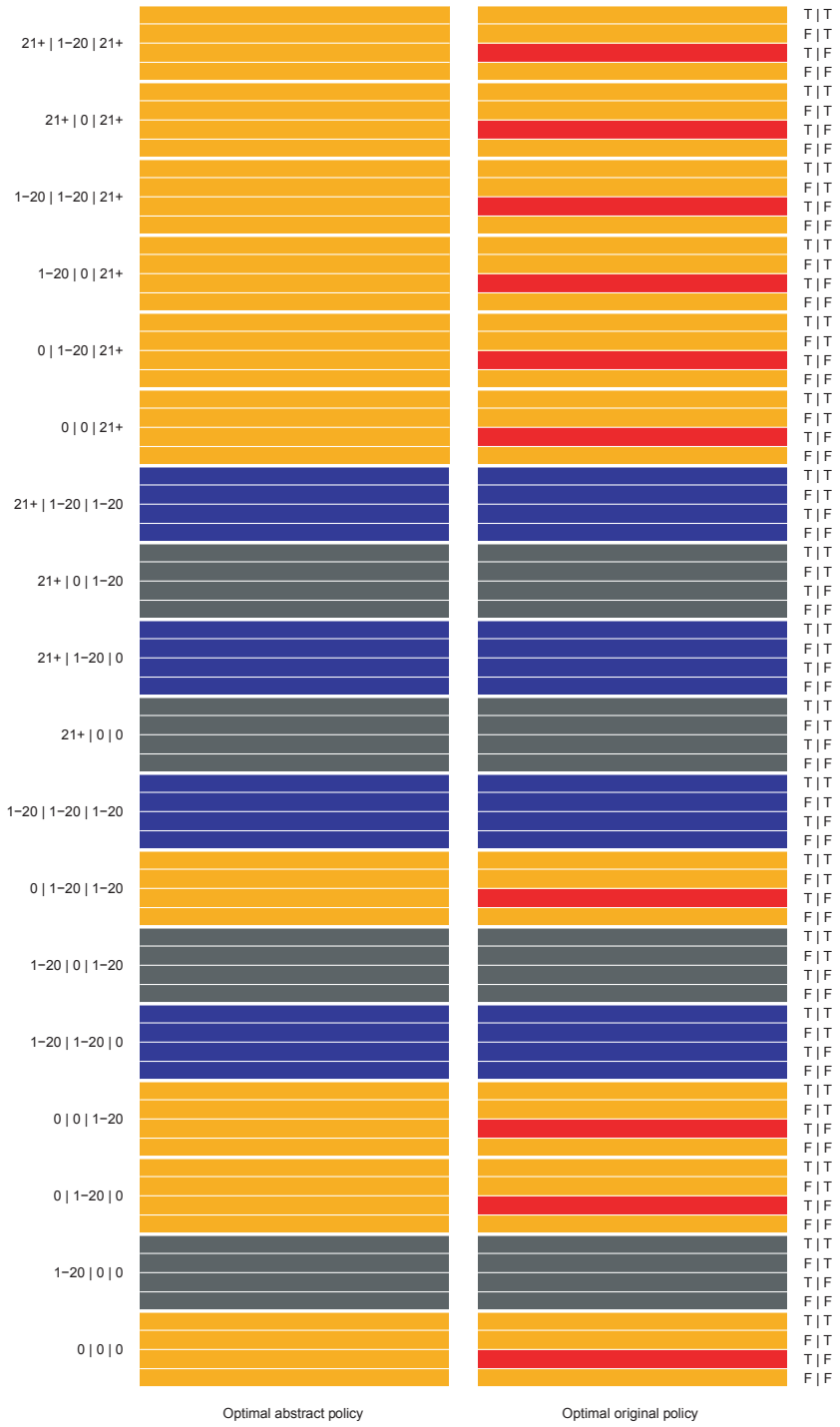

Fig. 1. Optimal abstract and original policies for a single stage of the species reintroduction problem. The states of the system are described by the different assignments to the five state variables (in order): *Source Population Size*; *Captive Population Size*; *Target Population Size* (values shown on the far left axis); *Captive Infected* (T = True; F = False); *Wild Infected* (values shown on the far right axis). Colours indicate optimal decision: Grey = *Capture*; Blue = *Release*; Red = *Isolate*; and Orange = *Do Nothing*.

size. The abstraction mechanism ensures that, all else being equal, solution quality does not degrade with increases in the number of states. Like before, the *a priori* error bound represents a worse-case bound, but because the size of the state space kept us from finding the optimal solution, we have no true values to compare against. It is also impossible in this short paper to display the decision space for the entire problem; for simplicity, we only present results from a sample of states where the abstract policy is implemented in the original state space (Fig. 2).

Discussion

Most sequential decision-making problems in conservation can be viewed conceptually and modelled as a Markov decision process. There is a rapidly growing and increasingly complex literature that reflects this point. Most authors are quick to praise this optimization framework; yet, it would be difficult to find a paper that does not include a one-line caveat saying that the method will be limited (sometimes severely) by the curse of dimensionality. It is therefore surprising that such little

Table 3. Species reintroduction problem: comparing the results of the abstract and optimal policies, using 3 different reward functions. Emphasis is placed on the actual average value lost per stage by using the abstract policy in the original domain.

Abstract policy vs. optimal policy	Nominal reward	Uniform reward	Preferred reward
Number of errors in action choice	10 (13·89%)	16 (22·22%)	20 (27·78%)
a priori Error bound (max. value lost per stage)*	10·00 10·00%	25·00 25·00%	40·00 40·00%
Actual max. value lost per stage*	5·90 5·90%	14·76 14·76%	23·62 23·62%
Actual average value lost per stage†	**0·61** **2·33%**	**1·85** **4·23%**	**3·81** **8·04%**
Average error in value of state†	4·84 9·93%	11·10 22·21%	16·67 43·85%
Standard deviation	1·21	3·52	6·33

*Percentages expressed as a function of maximum value.
†Percentages expressed as the average of the deviance (average value lost; average error) divided by the true optimal value for each state.

Table 4. Comparing several possible abstractions for the landscape restoration problem. The *a priori* error bound is calculated with respect to the reward definition provided in Table 3. The abstraction selected in the paper is shown in bold.

No. of state variables in abstract MDP	*A priori* error bound (%)	# States	% Reduction in state space
5	105 (48·837)	32	99·997
6	90 (41·860)	64	99·994
7	75 (34·883)	128	99·987
8	60 (27·907)	256	99·975
9	45 (20·930)	512	99·951
10	**30 (13·953)**	**1024**	**99·902**
11	25 (11·627)	2048	99·804
12	20 (9·302)	4096	99·609
13	15 (6·976)	8192	99·518
14	10 (4·651)	16 384	98·843
15	5 (2·325)	32 768	96·875
16	4 (1·860)	65 536	93·750
17	3 (1·395)	131 072	0·875
18	2 (0·093)	262 144	0·750
19	1 (0·046)	524 288	0·500

MDP, Markov decision processes.

attention has been given to approximation methods (but see Nicol *et al*. 2010; Chades *et al*. 2011; Nicol & Chades 2011). The abstraction algorithm of Dearden & Boutilier (1997) provides a mechanism to simplify MDPs with large state spaces. Using a hypothetical species reintroduction problem, we have demonstrated the performance of the algorithm against the optimal dynamic programming solution. This is the first time that an aggregation method of this kind has been applied to problems in a conservation setting. The algorithm is potentially useful in complex decision-making contexts, and our efforts will hopefully draw attention to methods being developed in the artificial intelligence community. We hope to provide a discussion of the method and results as they relate, generally speaking, to the problem of approximating solutions to large MDPs when standard action descriptions are no longer practical. In doing so, we discuss limitations of and alternatives to the approach and suggest directions for extending this work.

A number of sequential decision-making problems may prove amenable to the abstraction algorithm demonstrated here. Characteristics that will allow good abstractions include the presence of variables that are only marginally relevant to the problem, a multiattribute reward function in which the goals of the problem may by achieved or maintained independently, and sub-goals whose contribution to the reward function are larger than those of other subgoals (Dearden & Boutilier 1997). Problems with these characteristics could exist in any part of conservation planning, but perhaps most significant is the area of spatial prioritization. Applications of dynamic programming to landscape reconstruction (e.g. Chauvenet *et al*. 2010) and reserve selection (e.g. Costello & Polasky 2004), for example, have usually been limited to problems with less than ten sites. The abstraction algorithm could easily apply to any such problem where sites are prioritized with a scoring system. Other areas include metapopulation management where the number of patches is large (e.g. Westphal *et al*. 2003), and local or regional resource allocation and scheduling problems where, again, the number of sites is greater than 10–15. This research also has implications for ecological disciplines outside of conservation planning, including behavioural and evolutionary ecology, biocontrol in agriculture and optimal harvesting.

We have stressed throughout that the utility of an abstraction is a function of the feasibility of the specification. As the feasibility of the specification is a function of the size of the state space, and the size of the state space is exponential in the number of state variables, any reduction in the relevant set will result in a dramatic reduction in the size of the state space. In the landscape restoration problem, we were able to reduce the number of states by 99·9% (1024 vs. 1 048 576) and reduce the size of the transition matrices by 99·99% (1 048 576 vs. 1 099 511 627 776).

This reduction in the size of the state space comes at the cost of potentially generating suboptimal policies. However, the extent to which the abstract policy agrees with the optimal policy provides a measure of the quality of the approximation. In particular, we are interested in the number of states where the abstract action matches the optimal action. In the reintroduction problem, we were able to reduce the number of states by 75% (18 vs. 72) reduce the size of the transition matrices by almost 94% (324 vs. 5184), and the abstract action matched the optimal action more than 86% of the time. More importantly, the abstract policy was *never* wrong on the big decisions. These 'big' decisions were related to maintaining *Source Population* and establishing *Target Population*. Put another way, errors in action choice occurred only when *Capture* and *Release* were not optimal actions in the original MDP. We tried to break the abstraction using alternative definitions of the reward function and yes, performance did drop, but even under the 'worst-case' scenario (represented by the Preferred Reward alternative) the average value lost per stage was less

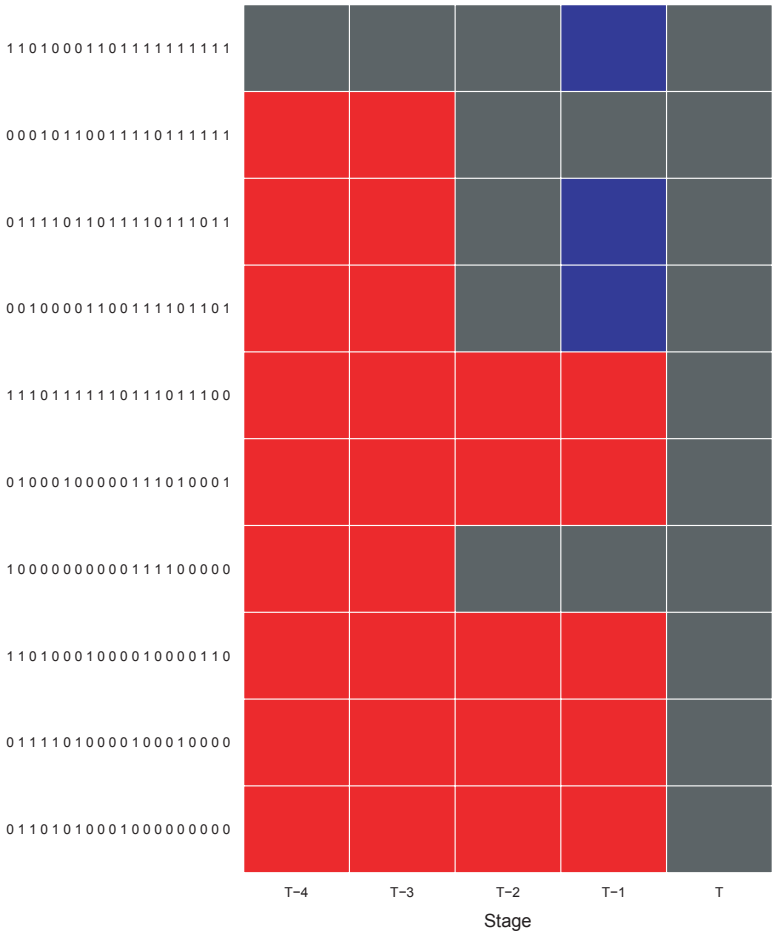

Fig. 2. A sample of the abstract policy in the original landscape restoration problem. The states of the system are described by the different assignments to the twenty state variables representing the twenty sites (site quality increases as you move from left to right). This policy was generated with the following probabilities of success given management: *P* (Success | *Top Two*) = 1·00; *P*Success | *Top Five*) = 0·65; *P*(Success | *Top Ten*) = 0·2. The annual probability of site degradation (without management) was set equal to 0·7. Colours indicate optimal decision: Grey = *Top Five*; Blue = *Top Two*; and Red = *Top Ten*.

than 9%, and the number of optimal actions chosen by the abstract policy was 52 (of 72) or more than 72%.

Other state aggregation methods have been developed to deal with the curse of dimensionality. Uther & Veloso (1998) introduced an automated method (Continuous U-Tree) that takes a continuous, or ordered discrete state space and splits it to form a discretization (a continuous MDP can be thought of as an MDP with an infinite number of states; for an application in conservation, see Nicol & Chades 2012). Continuous U-Tree only adds states that are necessary (based on a statistical measure) to maintain the optimal solution. Significance is based on a two-sample Komogorov–Smirnov test. Boutilier, Dearden & Goldszmidt (2000) introduced an exact algorithm that uses a dynamic Bayesian network implemented as a decision diagram to create an abstract model where states with the same transition and reward functions (under a fixed policy) are grouped together (for an application in conservation, see Chades *et al.* 2011). Givan, Dean & Greig (2003) called this a form of bisimulation, and several adaptations have been proposed. Ferns, Panangaden & Precup (2004), for example, developed a statistical metric to determine the similarity between two states' transition probabilities, which is then combined with information from the reward function to determine aggregation.

The optimality provided by the exact aggregation methods described previously is appealing; however, there are at least four disadvantages to consider. First, exact aggregation methods like those presented by Boutilier, Dearden & Goldszmidt (2000) require the state transition function for the entire problem to determine the aggregation scheme, whereas with the abstraction algorithm presented here we can assign the transition probability for any state in the cluster to the cluster itself (meaning we really only have to define the state transition function for the reduced problem). Second, exact aggregation methods require extensive effort upfront (e.g. building the dynamic Bayesian network) to determine which states have the same transition and reward functions. In some circumstances, this effort will most likely exceed that required to build and solve the MDP in a traditional manner. With the abstraction algorithm, the only required effort of this kind is building the stochastic effects list for each discriminant. Third, problems must have the proper structure for them to be an exact aggregation that actually simplifies the state space (Boutilier, Dearden & Goldszmidt 2000); for an aggregation to be exact, every state in the cluster of the reduced problem must behave the same as in the full problem (Li, Walsh & Littman 2006). This will not always occur, especially when each state has a different reward assigned to it or has a unique transition probability. Using the STRIPS-style, reward representation to determine the abstraction scheme ensures that we can always simplify the state space. Finally, exact aggregation methods

can be sensitive to changes in the parameters of the state transition function. This is because states are grouped when they have the same transition probability and/or reward. As a result, the aggregation may change or no longer be possible following even subtle adjustments in the parameters of the state transition function. The abstraction mechanism (and *a priori* error bound) operates the same regardless of how the transition function is parameterized.

We have introduced a new algorithm to conservation decision-making. The idea behind the abstraction algorithm is to capture the most important aspects of the original MDP, find an optimal policy over this reduced space and use this as an approximate solution to the original problem. The algorithm is not a widespread remedy for the curse of dimensionality, but it does advance our ability to construct approximately optimal policies in systems with large state spaces. We hope our work will stimulate additional interest in approximate optimization techniques in conservation planning.

Acknowledgements

T. Hefley, E. Blankenship and three anonymous reviewers provided their helpful comments on an earlier version of this manuscript. A. Schapaugh was funded by the University of Nebraska-Lincoln.

References

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

Bellman, R. (1961) *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey.

Bertsekas, D. (2007) *Dynamic Programming and Optimal Control*, vol. **II**, 3rd edn. Athena Scientific, Nashua, New Hampshire.

Bogich, T. & Shea, K. (2008) A state-dependent model for the optimal management of an invasive metapopulation. *Ecological Applications*, **18**, 748–761.

Boutilier, C., Dean, T. & Hanks, S. (1999) Decision theoretic planning: structural assumptions and comupational leverage. *Journal of Artificial Intelligence Research*, **11**, 1–94.

Boutilier, C., Dearden, R. & Goldszmidt, M. (2000) Stochastic dynamic programming with factored representations. *Artificial Intelligence*, **121**, 49–107.

Chades, I., Martin, T., Nicol, S., Burgman, M., Possingham, H.P. & Buckley, Y. (2011) General rules for managing and surveying networks of pests, diseases, and endangered species. *Proceedings of the National Academy of Sciences*, **108**, 8323–8328.

Chauvenet, A., Baxter, P., McDonald-Madden, E. & Possingham, H.P. (2010) Optimal allocation of conservation effort among subpopulations of a threatened species: How important is patch quality? *Ecological Applications*, **20**, 789–797.

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

Costello, C. & Polasky, .S. (2004) Dynamic reserve site selection. *Resources and Energy Economics*, **26**, 157–174.

Dearden, R. & Boutilier, C. (1997) Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, **89**, 219–283.

Ferns, N., Panangaden, P. & Precup, D. (2004) Metrics for finite Markov decision processes. In: *Proceedings of the Twenthieth Conference on Uncertainty in Artificial Intelligence*, pp. 162–169.

Fikes, R. & Nilsson, N. (1971) STRIPS. A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **2**, 189–208.

Givan, R., Dean, T. & Greig, M. (2003) Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, **147**, 163–223.

Kushmerick, N., Hanks, S. & Weld, D. (1994) An algorithm for probabilistic least-commitment planning. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1073–1078.

Li, L., Walsh, T.J. & Littman, M.L. 2006. Towards a unified theory of state abstraction for MDPs. In: *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pp. 531–539.

Nicol, S. & Chades, 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.

Nicol, S. & Chades, I. (2012) Which states matter? An application of an intelligent discretization method to solve a continuous POMDP in conservation biology *PLoS ONE*, **7**, e28993. doi:10.1371/journal.pone.0028993.

Nicol, S., Chades, I., Linke, S. & Possingham, H.P. (2010) Conservation decision-making in large state spaces. *Ecological Modelling*, **221**, 2531–2536.

Possingham, H.P., Andelman, S., Noon, B., Trombulak, S. & Pulliam, H., 2001. Making smart conservation decisions. *Conservation Biology: Research priorities for the next decade* (eds M.E. Soule & G.H. Orians), pp. 225–244. Island Press, Washington, District of Columbia, USA.

Powell, W. (2007) *Approximate Dynamic Programming: Solving the Curse of Dimensionality*. Wiley-Interscience, New York, New York.

Putterman, M. (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, New York.

R Development Core Team (2009) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Tenhumberg, B., Tyre, A.J., Shea, K. & Possingham, H.P. (2004) Linking wild and captive populations to maximize species persistence: optimal translocation strategies. *Conservation Biology*, **18**, 1304–1314.

Uther, W. & Veloso, M.M., 1998. Tree based discretization for continuous state space reinforcement learning. In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence AAAI-98*, pp. 769–794.

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.

Received 10 March 2012; accepted 24 July 2012
Handling Editor: Robert Freckleton