24th International Congress on Modelling and Simulation, Sydney, NSW, Australia, 5 to 10 December 2021
mssanz.org.au/modsim2021

Solving constrained K-Markov decision processes

Jonathan Ferrer-Mestres [a] and Iadine Chades [a]

[a] *Conservation Decisions Team, Land and Water, CSIRO*
Email: jonathan.ferrermestres@csiro.au

Abstract: Markov Decision Processes (MDPs) are convenient mathematical models used to solve sequential decision-making problems when the state of the system is completely observable and actions have an uncertain outcome. In human operated systems such as conservation of biodiversity and environmental management, solutions that cannot be understood or explained to human experts are unlikely to be accepted and applied in real world problems. To tackle this issue, domain experts must find a clever way of visualising and understanding solutions, for example, plotting graphs and performing scenario analysis. Previous work proposed to solve K-MDPs, i.e, given an original MDP and a parameter K, generate a reduced state space MDP with at most K states. State abstractions proved to reduce the complexity and increase the interpretability of MDP solutions and models. However, states aggregated according to algorithms that minimize the loss of performance may not have a meaning for human experts. To bridge the gap between artificial intelligence (AI)-designed state space and human experts, we address the challenge of including human preferences in the state aggregation process. Building on previous work, we define the problem of solving constrained K-MDPs, i.e, generate a reduced state space MDP with at most K states aggregated according to a set of constraints $\mathcal{C}$ specified by a user. We present a new algorithm and assess its performance on two computational sustainability case studies from the literature with one and two state variables respectively. Our results show that we can achieve a substantial reduction on the size of MDP solutions at a small cost of performance, by aggregating states according to user preferences. We have reduced the size of the state space of our case studies up to a 99.75%, with a small loss of performance. Our reduced constrained K-MDP models and solutions are compact and more interpretable. We hope our approach helps reducing the gap between artificial intelligence (AI) systems and human decision makers and increases the uptake of MDPs by improving interpretability and including users in the decision process.

Keywords: *Markov decision processes, interpretability, explainable AI, computational sustainability.*

1 INTRODUCTION

Markov Decision Processes (MDPs) have been used to tackle sequential decision making problems with full state observability and uncertainty on the action's outcome. Stochastic dynamic programming (SDP) and MDPs have been largely used to solve conservation decision problems (Memarzadeh et al. 2019), natural resource management (Pozzi et al. 2017) and behavioural ecology (Venner et al. 2006, Mangel et al. 1988). One of the identified barriers preventing a greater uptake of artificial intelligence (AI) tools in conservation decision domains is the complexity of understating MDP models and solutions (Tulloch et al. 2015). MDP models and policies with thousands of states and transitions are in practice difficult to understand for decision makers. This is especially critical in human operated systems, where experts decide whether or not to follow the recommendations of a system. For most of the decision making, people generally do not attempt to find or apply optimal solutions, but apply simple and well understood decision-making strategies (Reeson & Dunstall 2009). To address this issue, it is essential that models and solutions can be easily explained and interpreted to guide decision-making and increase trust in artificial intelligence systems. Better interpretability can be achieved by facilitating visualization of models and solutions (Chades et al. 2012), in form of graphs, matrices and look-up tables. The policy graph is one common representation of an MDP policy. A policy graph is a visual representation of a policy where nodes represent states and edges represent the optimal action to apply at each state and its probability of transitioning to any other or same state. Nevertheless, MDPs with $|S|$ states require $|S|$ nodes with at most $2^{|S|}$ edges. Large state and action space MDPs are complex to represent using policy graphs or look-up tables.

A potential solution is to reduce the size of the state space in order to produce simpler and easier to interpret models and solutions. State abstractions reduce the size of the state space by aggregating those states that are equal or similar giving a metric or a state abstraction function. State space reduction for MDPs have been explored in the past to solve large state space MDPs by grouping those states whose transition probabilities and rewards are similar (Dean & Givan 1997). This approach is known as model similarity. Later on, Li et al. (2006) presented a unified theory of state abstraction for MDPs and introduced different abstraction schemes. Abel et al. (2016) built on Litman's work and presented approximate state abstractions with performance guarantees. They introduced a collection of metrics also known as state abstraction functions. Those state abstraction functions allow an aggregation between two states if a given condition is satisfied and with a minimum loss of performance. Abel et al. (2018) presented transitive state abstractions, allowing a reduction in the computation time of state abstractions. However, in these approaches, the human operator does not decide how much the state space is reduced. Previous work showed how to solve K-MDPs (Ferrer-Mestres et al. 2020), i.e. given an MDP and a parameter K representing a constraint in the number of states, generate a reduced state MDP, called K-MDP, with at most K states. The authors presented a family of algorithms, based on binary search on a precision parameter and state abstraction functions. However, while the resulting K-MDP model and policies proved to reduce the complexity and increase the intepretability of solutions, the state aggregations are computed with the human operator not playing an active role on deciding how the aggregated states look. The K-MDP is computed according to an algorithm that minimizes the loss of performance between the optimal original MDP policy and the optimal reduced K-MDP policy.

We tackle the challenge of including a human operator in the loop, and we propose to solve constrained K-MDPs, i.e. Given an MDP M, a parameter K defining the maximum number of states and a set of constraints $\mathcal{C}$ defining the states that must be aggregated together, generate a reduced MDP called constrained K-MDP with at most K states that have been aggregated according user preferences. The aim of this work is to increase interpretability and explainability in MDP models and solutions in human operator domains by including the manager or expert feedback in the loop. While we lose performance guarantees, we modify the K-MDP according to what makes more sense for the user. The paper is organized as follows: Section 2 reviews MDPs and K-MDPs; Section 3 defines the problem of solving constrained K-MDPs and Section 4 presents an algorithm for solving constrained K-MDPs: (Enforced $\bar{\phi}_{a_d^*}K$-MDP). Finally, we evaluate our approach in two conservation decision problems and we discuss results and future research.

2 MDPs AND K-MDPs

Markov Decision Processes (MDPs) are fully observable, probabilistic state transition models where the decision-maker has complete information of the state of the system but is uncertain about the actions outcome (Puterman 2014). More formally, we define a finite MDP M as a tuple $\langle S, A, T, r, H, \gamma \rangle$, where: S is the fully observable state space; A is the set of actions the manager can choose to apply in a given state at each time step; T is the transition probability function describing the stochastic dynamics of the system; r is the reward function describing the benefits and cost of performing an action a at a particular state s; H is the planning horizon, and $\gamma \in [0, 1]$ is a discount factor relating future rewards and costs. A discount factor lower than one expresses that immediate rewards are more valuable than later ones.

The solution to an MDP is a function $\pi : S \rightarrow A$, called policy, that maps states into actions and maximises an optimisation objective. We can evaluate and rank the performance of different policies based on their expected values given an optimisation criterion. $V^{\pi}(s)$ is the expected value of executing policy π starting in state s. We denote π^* an optimal policy and V^* the optimal value function. Formally, we define the value of a policy, i.e. the value function as $V^{\pi}(s) = E^{\pi}[\sum_t^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]$ (Bellman 1957). The optimal policy π^* maximises this expected sum in any state: $\pi^* = \text{argmax}_{\pi}\, E[\sum_t^{\infty} \gamma^t r(s_t, \pi(s_t))|s_0 = s]$. Solving an MDP is polynomial in time (Papadimitriou & Tsitsiklis 1987).

2.1 K-MDPs

The problem of solving K-MDPs was introduced by Ferrer-Mestres et al. (2020).

Definition 1. *Given an MDP $M = \langle S, A, T, r, \gamma\rangle$, Let us define a K-MDP as a tuple $M_K = \langle S_K, A, T_K, r_K, \gamma, \phi\rangle$, where: S_K the abstract state space with $|S_K| \leq K$; A is the same set of management actions as in the original MDP model; $T_K : S_K \times A \times S_K \rightarrow [0, 1]$ is the abstract K-MDP probability transition function; $r_K : S_K \times A \rightarrow [0, R_{max}]$ the abstract reward function describing the benefits and costs of the reduced state space system; ϕ a function that maps a state s in the original MDP to a state s_K in the abstract K-MDP; γ is the discount factor as in the original MDP.*

An optimal solution for a K-MDP is a policy $\pi_K^* : S_K \rightarrow A$ that maximizes the expected sum of discounted rewards and can be calculated using exact MDP solvers.

3 Solving Constrained K-MDPs

Reducing the size of the state space increases the interpretability of both MDP models and solutions. K-MDPs are solved using a family of algorithms (Ferrer-Mestres et al. (2020)), but the resulting aggregated states are computed as a “ black ” box, without any feedback from the user. We now propose to include the human in the loop when aggregating states. Our approach requires an initial MDP, a parameter K defining the maximum number of states and a set of constraints $\mathcal{C}$ defining what states must be aggregated together. Formally,

Definition 2. *Given an MDP $M = \langle S, A, T, r, H, \gamma\rangle$, a constrained K-MDP is defined as a tuple $\bar{M}_K = \langle S_K, A, T_K, r_K, \gamma, \bar{\phi}, \mathcal{C}\rangle$ where:*

- $S_K = \{\bar{\phi}(s)|s \in S\}$ the abstract state space with $|S_K| \leq K$ and $\bar{\phi}$ a function that maps a state s in the original MDP to a state s_K in the abstract K-MDP. The inverse of function $\bar{\phi}^{-1}(s_K)$ maps an abstract state $s_K \in S_K$ to its constituent states in the original MDP.
- A is the same set of management actions as in the original MDP model.
- $T_K : S_K \times A \times S_K \rightarrow [0, 1]$ is the abstract K-MDP probability transition function providing the probability of being in state s'_K at time $t + 1$ given action a was implemented in state s_K at time t, $T_K(s_K, a, s'_K) = \sum_{s\in\bar{\phi}^{-1}(s_K)} \sum_{s'\in\bar{\phi}^{-1}(s'_K)} T(s, a, s')\omega(s)$, where the weights $\omega(s)$ represent a probability distribution over the original states that aggregate to an abstract state s_K (Abel et al. 2016):
$$\forall s_K \in S_K, \left(\sum_{s\in\bar{\phi}^{-1}(s_K)} \omega(s)\right) = 1,\ \omega(s) \in [0, 1].$$
- $r_K : S_K \times A \rightarrow [0, R_{max}]$ the abstract reward function defined as a weighted sum over the original states $r_K(s_K, a) = \sum_{s\in\bar{\phi}^{-1}(s_K)} r(s, a)\omega(s)$.
- $\mathcal{C}$ is a set of constraint with each constraint defining a a set of states that are forced to be aggregated together. i.e. $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ where $c_i = \{s_1, s_2, \ldots, s_j\}$.

The optimal solution for a constrained K-MDP is a constrained policy $\bar{\pi}_K^* : S_K \rightarrow A$ that maximizes the expected sum of discounted rewards. The optimal policy can be applied to the original MDP using the mapping function $\bar{\phi}$, with the associated value function $V_{\bar{\phi}}^{\bar{\pi}_K^*}(s) = E[\sum_{t=0}^{t=H} \gamma^t r(s_t, \bar{\pi}_K^*(\bar{\phi}(s_t)))|s_0 = s]$.

Essentially, $V_{\bar{\phi}}^{\bar{\pi}_K^*}$ represents the performance of the constrained K-MDP policy $\bar{\pi}_K^*$ when applied to the original MDP problem. In other words, we evaluate the performance of the reduced policy on the original MDP model. We formulate the problem of finding the best reduced state space S_K as a gap minimization problem between the original MDP value function and the abstract constrained K-MDP value function $gap^* = \min_{S_K \in \mathcal{P}(S),\, |S_K| \leq K} \max_{s \in S} [V^{\pi^*}(s) - V_{\bar{\phi}}^{\bar{\pi}_K^*}(s)]$. The optimal gap gap^* is the difference between the optimal MDP value function V^* and the optimal associated constrained K-MDP value function $V_{\bar{\phi}}^{\bar{\pi}_K^*}$. Our algorithms will aim to define constrained K-MDPs that minimise this gap (i.e. error). To do this, we first define a state abstraction function, i.e. rule to group states together. This is a challenging procedure, because the quality of the state grouping rules will influence the performance of the reduced MDP. Fundamentally, the constrained K-MDPs definition adds a set of constraints on the states that forces them to be aggregated together, to the definition of a K-MDP provided by Ferrer-Mestres et al. (2020).

4 Proposed algorithms to solve constrained K-MDPs

We now propose an algorithm that take a set of constraints $\mathcal{C}$ defined by the manager and return a constrained K-MDP, where, although the optimal constrained K-MDP policy does not minimize the loss of performance, we gain on interpretability by including the manager in the loop. We first adapt the approximate transitive $\phi_{a_d^*}$ abstraction function and the $\phi_{a_d^*}K$-MDP algorithm. For clarity, we now call the state abstraction function, $\bar{\phi}_{a_d^*}$, and the algorithm, $\bar{\phi}_{a_d^*}K$-MDP. For any pair of states s_1 and s_2, the approximate transitive abstraction function $\bar{\phi}_{a_d^*}$ satisfies

$$\bar{\phi}_{a_d^*}(s_1) = \bar{\phi}_{a_d^*}(s_2) \implies \bar{a}_{s_1} = \bar{a}_{s_2} \quad \& \quad \left\lceil \frac{\bar{V}^*(s_1)}{d} \right\rceil = \left\lceil \frac{\bar{V}^*(s_2)}{d} \right\rceil, \tag{1}$$

where d takes a value between 0 and VMAX, $\bar{a}_{s_1}$ and $\bar{a}_{s_2}$ are management actions, and $\bar{V}^*$ is the constrained value function. According to Eq. 1, two states s_1 and s_2 can be aggregated if they belong to the same bin and they have the same feasible actions. Our proposed Alg. 1 takes as input an MDP M, a constraint on the number of states K, a constrained value function $\bar{V}^*$, a constrained policy $\bar{\pi}^*$ and a precision parameter p_{target}. The algorithm performs a binary search on d and sorts the states into bins.

Algorithm 1 $\bar{\phi}_{a_d^*}K$-MDP

```
Require: $M, K \geq 1, \bar{V}^*, \bar{\pi}^*, p_{target}, \bar{\phi}$
 1: $d^+ = $ VMAX
 2: $d^- = 0$
 3: repeat
 4:     $p = d^+ - d^-$
 5:     $d = d^- + \frac{d^+ - d^-}{2}$
 6:     for $s \in S$ do
 7:         $bindings \leftarrow [\lceil \frac{\bar{V}^*(s)}{d} \rceil, \bar{\pi}^*(s)]$
 8:     end for
 9:     $S_K \leftarrow unique(bindings)$
10:     if $|S_K| \leq K$ then $d^+ \leftarrow d$
11:     else $d^- \leftarrow d$
12: until $p < p_{target}$
13: $\bar{M}_K \leftarrow$ BUILD-K-MDP$(M, S_K, \phi)$
14: return $\bar{M}_K$
```

Algorithm 2 *Enforced* $\bar{\phi}_{a_d^*}K$*-MDP*

```
Require: $M, K, \mathcal{C}, p_{target}, \bar{\phi}$
 1: $\bar{S} \leftarrow S$
 2: for $c \in \mathcal{C}$ do
 3:     $\bar{s}_c = aggregate\_states(c)$
 4:     $\bar{S} = \bar{S} - \{c\}$
 5:     $\bar{S} = \bar{S} \bigcup \bar{s}_c$
 6: end for
 7: $\bar{M} \leftarrow$ BUILD-K-MDP$(M, \bar{S}, \bar{\phi})$
 8: $\bar{\pi}^*, \bar{V}^* \leftarrow$SOLVE-MDP$(\bar{M})$
 9: for $\bar{s}_c \in \bar{S}$ do
10:     $\bar{\pi}^*(\bar{s}_c) \leftarrow \bar{a}_c$
11:     $\bar{V}^*(\bar{s}_c) \leftarrow \bar{v}_c$
12: end for
13: $\bar{M}_K \leftarrow \bar{\phi}_{a_d^*}K$-MDP$(M, K, \bar{V}^*, \bar{\pi}^*, p_{target}, \bar{\phi})$
14: return $\bar{M}_K$
```

4.1 Enforced $\bar{\phi}_{a_d^*}K$-MDP

Alg. 2 enforces the states specified in each constraint to be aggregated together in an independent bin, with any other state not specified in the corresponding constraint. In other words, given a set of constraints $\mathcal{C}$ and $c_i \in \mathcal{C}$, then $\forall s \in c_i, \bar{\phi}(s) = s_{K_i}$ and $\forall s \notin c_i, \bar{\phi}(s) \neq s_{K_i}$. Alg. 2 takes as input an MDP model M, a parameter K denoting a maximum number of states for the constrained K-MDP, a set of constraints $\mathcal{C}$, a precision value p_{target} and an approximate state abstraction function $\bar{\phi}_{a_d^*}$. The algorithm aggregates all states specified in a given constraint c into an abstract constrained state $\bar{s}_c$ (line 3) and it builds a state space of constrained aggregated states $\bar{S}$ (line 5). It then builds an MDP $\bar{M}$, with the set of constrained abstract states (line 7).

Then, the procedure runs the $\bar{\phi}_{a_d^*}K$-MDP (line 13) but with some modifications to ensure that no other state is aggregated to the abstract state defined by the constraint. To achieve this, the Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm enforces each constrained abstract state $\bar{s}_c$ to have a dummy optimal action $\bar{a}_c$ (line 10) and a dummy value $\bar{v}_c$ (line 11). This will ensure that the constrained abstract state is not aggregated with any other state.

5 Experimental Results

We report next the empirical results of our Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm when running on two different computational case studies. All algorithms have been implemented with MATLAB R2020a and we solved the MDPs and resulting constrained K-MDPs using the MDPtoolbox (Chadès et al. 2014). The discount factor for all our problems is $\gamma = 0.96$ and the precision parameter has a value of $p_{target} = 0.0001$.

Control of a population: Wolf Culling. Our first case study seeks to maximize the population of European wolves while providing that the population remains between a maximum (N_{max}) and a minimum (N_{min}) number of individuals (Marescot et al. 2013). There is one state variable X_t denoting the number of individuals at a given time t and a total number of $|S| = 2000$ states representing the population of wolves, with $s_1 = 0$ individuals, $s_2 = 1$ individual, $\ldots$. Managers can choose between $|A| = 11$ actions designating the culling rate from 0% to 100% with a discretization of 10%. Rewards increase with the abundance of individuals when the current population is within N_{min} and N_{max}. The population persists between $N_{min} = 500$ and $N_{max} = 1000$ individuals. Running a plain $\bar{\phi}_{a_d^*}K$-MDP algorithm with $K = 9$, we obtain a gap of 0%, with a K-MDP where states have been contiguously and linearly aggregated according to the population of individuals (Table 1). Table 1 shows the results of running the Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm. Each constraint is defined as an interval of the population of individuals whose representative states must be aggregated together. For example, constraint $c = [0, 799]$ means that all the states $s \in S$, representing a number of individuals between 0 and 799, must be aggregated together in a single abstract state s_K. In this example, this corresponds to states from s_0 to s_{800}. For the first and second instances, we force the constrained K-MDPs to have $K = 3$ and $K = 4$ constrained abstract states, producing policies with 20% and 33.4% of gap respectively. The optimal constrained policy shows the optimal harvest rate for each group of states. For the third instance, we define 5 constraints and K=5, and our Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm computes a constrained K-MDP with 15.7% of gap. The optimal constrained policy is to perform 0% culling when the number of individuals is less than 800, 40% culling when the number of individuals is between 800 and 1199, 50% when the population ranges from 1200 to 1599 individuals and 60% culling when the population is higher than 1600 individuals. We reduced the original MDP from 2000 states, to first, $K = 9$ abstract states, with 0.0% gap, and then, to $K = 3$ states, and according to user preferences, running the Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm. Fig. 1 shows the policy graphs for each instance. Each node represents a range of individuals, according to each defined constraint, and edges represent the optimal action and its probability of transitioning to other actions. The original optimal policy would have required 3000 nodes with at most 2^{3000} edges instead of 5 nodes with at most 2^5 edges for $K = 5$.

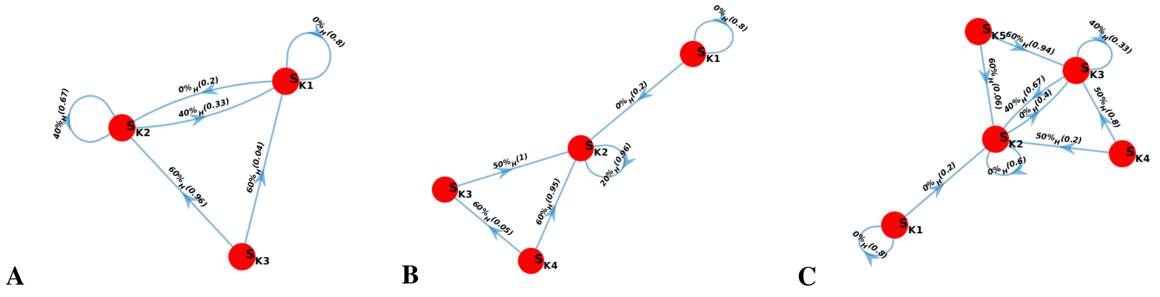

Figure 1. Constrained K-MDP policy graphs for the Wolf culling problem.

Recovering two endangered species: Sea Otter and Northern Abalone. Our second case study was described by Chades et al. (2012). The objective is to maximize the abundance of two endangered species, the sea otter (Enhydra lutris kenyoni) and the northern abalone (Haliotis kamtschatkana). Two state variables, X_t and Y_t, denote the abundance of sea otter and the density of northern abalone at a given time t. This problem

Table 1. Performance of the Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm. First column presents the case study and the result of running a plain $\bar{\phi}_{a_d^*}K$-MDP algorithm. Column 2 shows the constraints. Column 3 the value of K. Column 4 reports the gap. The rightmost column shows the constrained optimal policy for each instance.

Case study	Constraints	K	error (%)	$\bar{\pi}_K^*$	Time (sec.)
Wolf culling $\lvert S\rvert = 2000$, $\lvert A\rvert = 11$ $K = 9$ error= 0.0%	$c_1 = [0, 799]$ $c_2 = [800, 1499]$ $c_3 = [1500, 1999]$	3	20%	0% culling 40% culling 60% culling	1.92
	$c_1 = [0, 499]$ $c_2 = [500, 999]$ $c_3 = [1000, 1499]$ $c_4 = [1500, 1999]$	4	33.4%	0% culling 20% culling 50% culling 60% culling	1.74
	$c_1 = [0, 399]$ $c_2 = [400, 799]$ $c_3 = [800, 1199]$ $c_4 = [1200, 1599]$ $c_5 = [1600, 1999]$	5	15.7%	0% culling 0% culling 40% culling 50% culling 60% culling	1.75
Sea Otter and Northern Abalone $\lvert S\rvert = 819$, $\lvert A\rvert = 4$ $K = 5$ error = 1%	$c_1 = \{so :< 200\}$ $c_2 = \{so : [200, 4000]\}$	2	9.46%	Introduction Anti poaching	0.13
	$c_1 = \{so :< 200\}$ $c_2 = \{so : [200, 3500]\}$ $c_3 = \{so : [3500, 4000]\}$	3	1.65%	Introduction Antipoaching Half control	0.18
	$c_1 = \{na :< 0.2 \text{ AND } so :< 500\}$ $c_2 = \{na :< 0.2 \text{ AND } so : [500, 3000]\}$ $c_3 = \{na :< 0.2 \text{ AND } so :> 3000\}$	5	25%	Introduction Antipoaching Control Antipoaching Antipoaching	0.24

has been modeled as an MDP with 819 states representing the population of both species, and 4 actions: Introducing otters (I), anti poaching (AP), control sea otters (C) and one half anti poaching and one half control sea otters (H). We run our $\bar{\phi}_{a_d^*}K$-MDP algorithm and we reduce the state space up to $K = 5$ with a gap of only 1%. The second half of Table 1 shows the results of running our Enforced $\bar{\phi}_{a_d^*}K$-MDP algorithm. For the first instance, we define K =2 and two constraints on the abundance of sea otter; one denoting the abundance being smaller than 200, and the other one denoting the abundance ranging between 200 and 4000, independently on the density of northern abalone. The optimal constrained policy is to introduce sea otters when its abundance is less than 200, and anti poaching measures in any other case. The constrained K-MDP has a gap of only 9.46%. For the second instance, we define $K = 3$ and 3 constraints defining those states where the abundance of sea otter is i) less than 200, ii) between 200 and 3500 and iii) between 3500 and 4000. The optimal constrained policy is to introduce sea otters when the abundance is small (less than 200), anti poaching measures when the abundance is between 200 and 3500 and half control and half anti poaching when the abundance of sea otter is higher than 3500. The resulting constrained K-MDP has an gap of only 1.65%. For the last instance we define 3 constraints and a value of $K = 5$, letting the states that are not defined in the constraints to be aggregated according to the algorithm. The first constraint enforces to group those states where the density of northern abalone is lower than 0.2 and the abundance of sea otter is smaller than 500; in the second constraint, we group states when the density of northern abalone is below 0.2 and the abundance of sea otter is between 500 and 3000; and, the last constraint arranges states with a density of northern abalone below 0.2 and an abundance of sea otter greater than 3000. Interestingly, the resulting constrained K-MDP with $K = 5$ has a gap of 25%. Fig. 2 shows the optimal constrained K-MDP policy graphs with 2, 3 and 5 states respectively. The original policy would have required 819 nodes with at most 2^{819} edges instead of 2 nodes with at most 2^2 edges.

6 CONCLUSIONS

Motivated by the need to provide easier to interpret models and policies for MDPs, and to increase the uptake of artificial intelligence in human operated domains, we proposed to include the human in the loop, and we solve constrained K-MDPs. Our approach finds the best possible policy of size K while grouping together those states defined in each constraint. We have shown how users can define constraints to reduce the size of

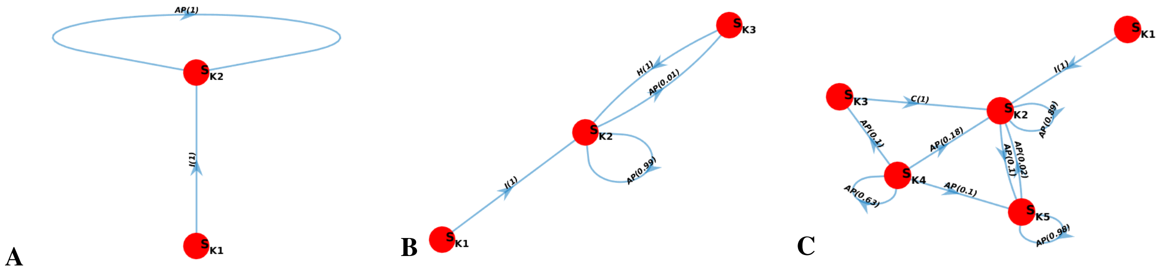

Figure 2. Constrained K-MDP policy graphs for the sea otter and northern abalone problem.

MDP models and policies to a few meaningful states, while retaining most of its value. For the wolf culling problem, we have reduced the size of the policy graph from 2000 states to 5 states with a gap of 15.7%. For the sea otter and northern abalone problem we have reduced the size of the policy graph from 819 states to 3 states with a gap of only 1.65%. This represents a considerable reduction of the state space at a small performance cost. The resulting constrained K-MDP policies are consistent with the original optimal policies. Modelers can choose how much they want to reduce MDP models and how they want to group these states to make models and policies more compact and interpretable. Future work will study how different choices of constraints for a same value of K, affect the computation of the constrained K-MDP.

Acknowledgements

The authors would like to thank Dr. Andrew Reeson who gave us the idea of including humans in the loop.

References

Abel, D., Arumugam, D., Lehnert, L. & Littman, M. (2018), State abstractions for lifelong reinforcement learning, *in* 'ICML'.

Abel, D., Hershkowitz, D. & Littman, M. (2016), Near optimal behavior via approximate state abstraction, *in* 'In ACML', Vol. 48 of *Proceedings of Machine Learning Research*, PMLR.

Bellman, R. (1957), *Dynamic programming*, Princeton Univeristy Press, John Wiley & Sons.

Chadès, I., Chapron, G., Cros, M.-J., Garcia, F. & Sabbadin, R. (2014), 'Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems', *Ecography* **37**(9).

Chades, I., Curtis, J. M. R. & Martin, T. G. (2012), 'Setting realistic recovery targets for two interacting endangered species, sea otter and northern abalone', *Conservation Biology* **26**(6).

Dean, T. & Givan, R. (1997), Model minimization in Markov decision processes, *in* 'AAAI/IAAI'.

Ferrer-Mestres, J., Dietterich, T. G., Buffet, O. & Chadès, I. (2020), Solving k-mdps, *in* 'ICAPS', Vol. 30.

Li, L., Walsh, T. J. & Littman, M. L. (2006), Towards a unified theory of state abstraction for mdps., *in* 'ISAIM'.

Mangel, M., Clark, C. W. et al. (1988), *Dynamic modeling in behavioral ecology*, Princeton University Press.

Marescot, L., Chapron, G., Chades, I., Fackler, P. L., Duchamp, C., Marboutin, E. & Gimenez, O. (2013), 'Complex decisions made simple: a primer on stochastic dynamic programming', *MEE* **4**(9).

Memarzadeh, M., Britten, G. L., Worm, B. & Boettiger, C. (2019), 'Rebuilding global fisheries under uncertainty', *Proceedings of the National Academy of Sciences* **116**(32).

Papadimitriou, C. H. & Tsitsiklis, J. N. (1987), 'The complexity of markov decision processes', *Mathematics of operations research* **12**(3).

Pozzi, M., Memarzadeh, M. & Klima, K. (2017), 'Hidden-model processes for adaptive management under uncertain climate change', *Journal of Infrastructure Systems* **23**(4).

Puterman, M. L. (2014), *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons.

Reeson, A. & Dunstall, S. (2009), 'Behavioural economics and complex decision-making', *Victoria: CSIRO* .

Tulloch, V. J., Tulloch, A. I., Visconti, P., Halpern, B. S., Watson, J. E., Evans, M. C., Auerbach, N. A., Barnes, M., Beger, M., Chadès, I. et al. (2015), 'Why do we map threats? linking threat mapping with actions to make better conservation decisions', *Frontiers in Ecology and the Environment* **13**(2).

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**(4).