Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Solving *K*-MDPs

Jonathan Ferrer-Mestres,[1] **Thomas G. Dietterich,**[2] **Olivier Buffet,**[3] **Iadine Chadès**[1]
[1]CSIRO, [2]Oregon State University, [3]INRIA
{jonathan.ferrermestres, iadine.chades}@csiro.au, tgd@cs.orst.edu, olivier.buffet@loria.fr

Abstract

Markov Decision Processes (MDPs) are employed to model sequential decision-making problems under uncertainty. Traditionally, algorithms to solve MDPs have focused on solving large state or action spaces. With increasing applications of MDPs to human-operated domains such as conservation of biodiversity and health, developing easy-to-interpret solutions is of paramount importance to increase uptake of MDP policies. Here, we define the problem of solving K-MDPs, i.e., given an original MDP and a constraint on the number of states (K), generate a reduced state space MDP that minimizes the difference between the original optimal MDP value function and the reduced optimal K-MDP value function. Building on existing non-transitive and transitive approximate state abstraction functions, we propose a family of three algorithms based on binary search with sub-optimality bounded polynomially in a precision parameter: $\phi_{Q^*_\epsilon} K$-MDP-ILP, $\phi_{Q^*_d} K$-MDP and $\phi_{a^*_d} K$-MDP. We compare these algorithms to a greedy algorithm ($\phi_{Q^*_\epsilon}$ Greedy K-MDP) and clustering approach (k-means++ K-MDP). On randomly generated MDPs and two computational sustainability MDPs, $\phi_{a^*_d} K$-MDP outperformed all algorithms when it could find a feasible solution. While numerous state abstraction problems have been proposed in the literature, this is the first time that the general problem of solving K-MDPs is suggested. We hope that our work will generate future research aiming at increasing the interpretability of MDP policies in human-operated domains.

Introduction

Markov decision processes (MDPs) are a convenient mathematical model for tackling sequential decision-making problems under uncertainty (Puterman 2014). MDPs have been applied to help recover populations of threatened species under limited resources, to control invasive species, to manage fisheries, to perform adaptive management of natural resources, and to test behavioral ecology theories (Marescot et al. 2013). These domains are human-operated systems, where MDP policies provide recommendations to humans rather than applied in an autonomous context such as in robotics. Policies computed for MDPs with thousands of states are in practice difficult to understand by humans. In human-operated systems, it is crucial that policies can be interpreted and explained to guide decision-making (Petrik and Luss 2016).

Explainable artificial intelligence (XAI), also know as the interpretability problem, can be divided into a) Explainability problems: generating decisions in which one of the criteria is how well a human could understand these decisions (our approach); b) Explanation problems: explicitly explaining decisions to humans. Emerging research on XAI has mostly focused on machine learning models rather than decision problems (Vellido, Martín-Guerrero, and Lisboa 2012; Guidotti et al. 2019). Work on interpretability for decision-making remains seldom (Dujardin, Dietterich, and Chadès 2015; 2017; Lakkaraju and Rudin 2017; Petrik and Luss 2016; Bertram and Wei 2018). We note recent interest in planning (XAIP) to generate explanations for planner solutions (Chakraborti et al. 2019; Krarup et al. 2019).

Modelling problems as Factored MDPs can help interpreting the solutions up to a point (Vianna, Sanner, and De Barros 2013). We found that when the number of state variables becomes too large (> 10), factored policies become too complex for experts in our application domains (e.g. http://iadine-chades.org/iacrc/network2/#). Here, we remained general and did not take advantage of factored properties.

Motivated by our interactions with computational sustainability experts puzzled with MDP policies, we add to these emerging XAI efforts and propose to increase the interpretability of MDPs. Inspired by work on solving N-POMDPs (Dujardin, Dietterich, and Chadès 2017; 2015), where N defines the maximum size of any admissible policy represented by a set of α-vectors, we propose to solve K-MDPs, i.e., to find the best MDP with at most K states, while taking advantage of leveraging on the state abstraction literature to develop new algorithms (Li, Walsh, and Littman 2006; Abel, Hershkowitz, and Littman 2016; Abel et al. 2018). State abstraction approaches aim to reduce the size of large state spaces by aggregating those states which are similar or equivalent given an abstraction function or a metric. In other words, a state abstraction function maps states into clusters of states such that solvers can compute a policy over those clusters. Finding good state abstraction functions has been the focus of many papers (Dearden and Boutilier 1997; Singh, Jaakkola, and Jordan 1995; Andre and Russell 2002;

Pineau, Gordon, and Thrun 2002; Dietterich 2000; Jong and Stone 2005). Of particular interest are state abstraction functions that minimize the loss of performance. Exact state abstraction functions aggregate states that are exactly equal given a metric (Li, Walsh, and Littman 2006; Dean and Givan 1997), but this very strong requirement limits our ability to aggregate states. More recently, approximate state abstraction functions have been proposed. These functions aggregate states that are closely similar given a measure (Abel, Hershkowitz, and Littman 2016; Abel et al. 2018). Interestingly, state abstraction approaches have mostly focused on finding the smallest state space given a loss of performance e.g. (Abel, Hershkowitz, and Littman 2016; Abel et al. 2017)—which does not solve the K-MDP problem as we seek to minimize the loss of performance given a maximum number of abstract states.

We first define the problem of providing easy-to-interpret MDP solutions as K-MDP problems. We build on approximate non-transitive and transitive state abstraction functions to develop new algorithms for finding the best K-MDP.

We present a set of experiments on randomly-generated MDPs and on two computational sustainability case studies—one involving the conservation of endangered species (Chades, Curtis, and Martin 2012) and one seeking to control an invasive mosquito species (Péron et al. 2017). Finally, we discuss future research directions.

MDPs

Markov Decision Processes (MDPs) provide a convenient model for representing sequential decision-making optimization problems when the decision maker has complete information about the current state of the system and dynamics are non-deterministic (Puterman 2014).

Formally, a finite MDP is specified as a tuple $\langle S, A, T, r, \gamma \rangle$, where:

- S is the set of perfectly observed states;
- A is the set of actions (or decisions) from which the manager needs to choose an action a at each time step;
- T is a probabilistic transition function describing the stochastic dynamics of the system; an element $T(s, a, s')$, for $s, s' \in S$ and $a \in A$, represents the probability of being in state s' at time $t+1$ given (s, a) at time t;
- $r : S \times A \rightarrow [0, R_{max}]$ is the reward function identifying the benefits or costs. For sake of simplicity, we assume that rewards are positive and bounded by R_{max} ;
- $\gamma \in [0, 1]$ is a discount factor.

The solution to an MDP is a function $\pi : S \rightarrow A$, called a policy, that specifies what action to take in each state. We can evaluate and rank policies based on their expected values given a criterion. The value $V^{\pi}(s)$ is the expected gain of implementing policy π starting in state s and continuing to a given time horizon. In the case of a discounted infinite horizon criterion, we have: $\forall s \in S, V^{\pi}(s) = E^{\pi}(\sum_t \gamma^t r_t | s_0 = s)$. We denote V^* the optimal value function that maximizes this expected sum in any state and π^* an optimal policy.

Most algorithms designed to solve MDPs, including the famous Value Iteration and Policy Iteration algorithms, rely on Bellman's dynamic programming equations (1958):

$$V(s) = \max_{a \in A(s)} [r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s')].$$

where $A(s)$ are the applicable actions in s. Solving an MDP is polynomial in time (Papadimitriou and Tsitsiklis 1987).

Our challenge is unusual. MDP solutions are rarely applied in our computational sustainability domains. We seek to increase interpretation of existing MDP solutions by generating compact MDPs rather than solving new MDPs.

K-MDPs

To increase the interpretability of MDP solutions, we propose to start with an initial MDP and find a simpler MDP called a K-MDP. We first introduce the problem of solving K-MDPs with minimum performance loss. We then define K-MDPs in the context of state abstraction.

General problem statement

Given an MDP $M = \langle S, A, T, r, \gamma \rangle$, a K-MDP $M_K = \langle S_K, A, T_K, r_K, \gamma, \phi \rangle$ is an MDP where S_K is a reduced state set of size at most K, A is the original set of actions, $T_K : S_K \times A \times S_K \rightarrow [0, 1]$ is the probability transition function, $r_K : S_K \times A \rightarrow [0, R_{max}]$, γ is the discount factor and ϕ is a mapping function from the original MDP state space S to the K-MDP state space S_K.

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. This policy can be applied to the original MDP by using the mapping function ϕ, with the associated value function $V_{\phi}^{\pi^*_K}$ where:

$$V_{\phi}^{\pi^*_K}(s) = E\left(\sum_{t=0}^{t=H} \gamma^t r(s_t, \pi^*_K(\phi(s_t))) | s_0 = s\right).$$

Essentially, $V_{\phi}^{\pi^*_K}$ represents the performance of policy π^*_K when applied to the original MDP problem.

We formulate the problem of finding the best reduced state space ($|S_K| \leq K$) as a gap minimization problem between the optimal MDP policy and the optimal K-MDP policy:

$$gap^* = \min_{S_K \in P(S), |S_K| \leq K} \max_{s \in S} [V^{\pi^*}(s) - V_{\phi}^{\pi^*_K}(s)], \quad (1)$$

where $P(S)$ is a power set of S.

In the remainder of the paper, we propose algorithms that minimize this gap. As a first step, we explore using state abstraction to reduce the state space. We acknowledge that alternative approaches could be used to generate K-MDPs.

K-MDPs using state abstraction

The goal of state abstraction approaches is to reduce the size of the MDP state space by aggregating and grouping those states which are similar given an abstraction function or metric (Dean and Givan 1997; Li, Walsh, and Littman 2006).

Given an MDP and a predicate $p(s_1, s_2)$ (a binary relation between states), a state abstraction function $\phi(s)$ satisfies:

$$(\phi(s_1) = \phi(s_2)) \implies p(s_1, s_2).$$

State abstraction functions can be exact or approximate. They can also have desirable properties such as transitivity:

$$p(s_1, s_2) \wedge p(s_2, s_3) \implies p(s_1, s_3).$$

We now provide a formal definition of a K-MDP in the specific context where state abstraction is used to solve the K-MDP problem.

Definition 1. *Given an MDP $M = \langle S, A, T, r, \gamma \rangle$, Let us define a K-MDP as a tuple $M_K = \langle S_K, A, T_K, r_K, \gamma, \phi \rangle$:*

- $S_K = \{\phi(s) | s \in S\}$ the abstract state space with $|S_K| \leq K$ and ϕ 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 $\phi^{-1}(s_K)$ maps an abstract state $s_K \in S_K$ to its constituent states in the original MDP.
- A is the same set of 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, $T_K(s_K, a, s'_K) = \sum_{s \in \phi^{-1}(s_K)} \sum_{s' \in \phi^{-1}(s'_K)} T(s, a, s')\omega(s)$, where the weights $\omega(s)$ represent a probability distribution over the original states that aggregate to an abstract state s_K:

$$\forall s_K \in S_K, \left(\sum_{s \in \phi^{-1}(s_K)} \omega(s) \right) = 1$$

  and $\omega(s) \in [0, 1]$ (Abel, Hershkowitz, and Littman 2016).
- $r_K : S_K \times A \rightarrow [0, R_{max}]$ the abstract reward function defined as a weighted sum over the original states $r_K(s_K, a) = \sum_{s \in \phi^{-1}(s_K)} r(s, a)\omega(s)$.

Unlike in (Abel, Hershkowitz, and Littman 2016), our definition adds a constraint on the number of states.

Proposed algorithms to solve K-MDPs

Building on the state abstraction function literature, we now propose a family of algorithms to solve K-MDPs.

All of our proposed algorithms need to build a K-MDP once the state space is reduced. We call this procedure *BUILD-K-MDP* (Alg. 1). Given an MDP M, an abstract state space S_K produced by one of our algorithms and ϕ, line 3 computes the weights of all states $s \in \phi^{-1}(s_K)$ (see Experimental Results). Lines 4 to 8 compute the corresponding reward function r_K and transition function T_K.

The $\phi_{Q^*_\epsilon} K$-MDP-ILP algorithm

Our first algorithm is based on a non-transitive approximate state abstraction function $\phi_{Q^*_\epsilon}$ proposed by Abel, Hershkowitz, and Littman (2016).

Algorithm 1 *BUILD-K-MDP*

Require: $M = \langle S, A, T, r, H, \gamma \rangle, S_K, \phi$
1: **for** $s_K \in S_K$ **do**
2: **for** $s \in \phi^{-1}(s_K)$ **do**
3: $\omega(s) \leftarrow$ *computeWeights*(ϕ, s_K)
4: $r_K(s_K, a) \leftarrow \sum_{s \in \phi^{-1}(s_K)} r(s, a)\omega(s)$
5: **for** $s'_K \in S_K$ **do**
6: **for** $a \in A$ **do**
7: $T_K(s_K, a, s'_K) \leftarrow$
8: $\sum_{s \in \phi^{-1}(s_K)} \sum_{s' \in \phi^{-1}(s'_K)} T(s, a, s')\omega(s)$
9: $M_K = \langle S_K, A, T_K, r_K, H, \gamma \rangle$
10: **return** M_K

Approximate non-transitive $\phi_{Q^*_\epsilon}$ abstraction function For any original states i, j, an approximate state abstraction function $\phi_{Q^*_\epsilon}$ satisfies:

$$\phi_{Q^*_\epsilon}(i) = \phi_{Q^*_\epsilon}(j) \implies \forall a \, |Q^*(i, a) - Q^*(j, a)| \leq \epsilon.$$

That we reformulate as:,

$$\phi_{Q^*_\epsilon}(i) = \phi_{Q^*_\epsilon}(j) \implies \max_a |Q^*(i, a) - Q^*(j, a)| \leq \epsilon.$$

Abel, Hershkowitz, and Littman (2016) show that an optimal policy derived with $\phi_{Q^*_\epsilon}$ applied to an original MDP has sub-optimality bounded polynomially in ϵ:

$$\forall s \in S, V^{\pi^*}(s) - V^{\pi^*_K}_{\phi_{Q^*_\epsilon}}(s) \leq \frac{2\epsilon R_{max}}{(1 - \gamma)^2}. \quad (2)$$

Because we seek to find a K-MDP that minimizes the gap (Eq. (1)), we can easily reformulate Eq. (2) as:

$$\max_{s \in S} V^{\pi^*}(s) - V^{\pi^*_K}_{\phi_{Q^*_\epsilon}}(s) \leq \frac{2\epsilon R_{max}}{(1 - \gamma)^2}. \quad (3)$$

Eq. (3) now provides a way of bounding the value loss in our K-MDP problem. Our proposed algorithm $\phi_{Q^*_\epsilon} K$-MDP-ILP exploits this theoretical result.

The $\phi_{Q^*_\epsilon} K$-MDP-ILP algorithm explained All states aggregated using $\phi_{Q^*_\epsilon}$ have values of Q^* within ϵ of each other. We define δ as a non-negative function such that $\delta(i, j) = \max_a |Q^*(i, a) - Q^*(j, a)|$, $i, j \in S$ and $a \in A$. Then, for each pair of states $i, j \in S$, if $\delta(i, j) \leq \epsilon$, then states i, j can be aggregated. We define the predicate $c_{i,j}$ returning true iff $\delta(i, j) \leq \epsilon$ holds.

Given all possible pairs $C = \{c_{i,j} | i, j \in S\}$ and ϵ, we want to find a state abstraction, if it exists, with at most K states. This is equivalent to partitioning the nodes of an undirected graph $G_C = \langle S, C \rangle$ into K cliques (Brigham and Dutton 1983), with the set of vertices defined as the states of the original MDP and the edges defined through C.

We propose a pure integer linear program ILP_{KMDP} with $|S|K + K$ 0-1 decision variables to address this problem as a node clique cover decision problem, which has been

proven NP-Complete (Karp 1972).

$$
\begin{aligned}
f: \quad & \min \sum_{k \in \mathcal{K}} z_k \\
\text{s.t.} \quad & \forall i \in S, \ \sum_{k \in \mathcal{K}} x_{ik} = 1 \\
& \forall i \in S, k \in \mathcal{K}, \ x_{ik} \leq z_k \\
& \forall k \in \mathcal{K}, i \in S, i < j \leq |S|, \ x_{ik} + x_{jk} - 1 \leq c_{ij} \\
& \forall i \in S, k \in \mathcal{K}, \ x_{ik} \in \{0, 1\} \\
& \forall k \in \mathcal{K}, \ z_k \in \{0, 1\} \\
& \qquad\qquad\qquad\qquad\qquad\qquad (ILP_{KMDP})
\end{aligned}
$$

We define $\mathcal{K}$ as the finite set $\{1, \ldots, K\}$. Given an undirected graph $G_C = \langle S, C \rangle$, ILP_{KMDP} encodes a state abstraction through at most K clusters:

1. Decision variable x_{ik} takes value 1 if state i belongs to clique $k \in \mathcal{K}$ and 0 otherwise;
2. Decision variable representing clique z_k takes value 1 if it contains at least one state and 0 otherwise;
3. A state $i \in S$ can only belong to one clique $k \in \mathcal{K}$;
4. Two states $i, j \in S$ that do not share an edge ($c_{ij} = 0$) cannot belong to the same clique k;
5. There are at most K cliques.

Remark. *Linear Program ILP_{KMDP} does not need an objective function since only the feasibility is checked. An objective function has been added to obtain a formal linear program.*

ILP_{KMDP} requires a graph $G_C = \langle S, C \rangle$ computed from an ϵ value. Algorithm $\phi_{Q_\epsilon^*}$ K-MDP performs a binary search to the decision version of the K-MDP problem using the $\phi_{Q_\epsilon^*}$ abstraction function and the linear program ILP_{KMDP}. Alg. 2 searches for the smallest ϵ value that allows a valid K-MDP state abstraction, with ϵ defined as in Eq. (3) and S_K a solution returned by ILP_{KMDP}.

Proposition 1. *Alg. 2 solves the K-MDP problem within precision p_{target} in a finite number of iterations. The optimal value function derived from the solution M_K, $V^{\pi_K^*}_{\phi_{Q_\epsilon^*}}$, has sub-optimality bounded polynomially in ϵ.*

Proof. The sub-optimality bounded polynomially in ϵ is a direct consequence of Eq. 3 and the use of $\delta(i, j)$ (line 6). The binary search algorithm applied to a continuous variable using an arbitrary precision p_{target} requires $\log(\frac{max_{i,j \in S} \delta(i,j)}{p_{target}})$ iterations and ILP_{KMDP} can be solved using Branch and Bound for a 0-1 integer linear program. There are $|S|K + K$ variables and $\frac{1}{2}K|S|^2 + \frac{1}{2}|S|K + |S|$ constraints. Therefore, the complexity of ILP_{KMDP} is $O(2^{K|S|})$. Alg. 2 has time complexity in $O(\log \frac{max_{i,j \in S} \delta(i,j)}{p_{target}} 2^{K|S|})$ due to binary search and branch and bound for solving ILP_{KMPD}. □

While this approach is mathematically elegant, it is limited by its computational complexity. We now propose faster algorithms.

Algorithm 2 $\phi_{Q_\epsilon^*}$ K-MDP-ILP

Require: $M, K \geq 1, p_{target}$
1: $\epsilon^- = 0, \epsilon^+ = max_{i,j \in S} \delta(i, j)$
2: **repeat**
3: $p = \epsilon^+ - \epsilon^-$
4: $\epsilon = \epsilon^- + \frac{\epsilon^+ - \epsilon^-}{2}$
5: **for** $i, j \in S$ **do**
6: **if** $\delta(i, j) \leq \epsilon$ **then**
7: $c_{i,j} = 1$
8: **else**
9: $c_{i,j} = 0$
10: $C = \{c_{i,j} | i, j \in S\}$
11: $S_K \leftarrow ILP_{KMDP}(G_C = \langle C, S \rangle, K)$
12: **if** $ILP_{KMDP}(G_C, K)$ has a solution **then**
13: $\epsilon^+ = \epsilon$
14: **else**
15: $\epsilon^- = \epsilon$
16: **until** $p < p_{target}$
17: $M_K \leftarrow$ *BUILD-K-MDP*(M, S_K, ϕ)
18: **return** M_K

The $\phi_{Q_d^*}$ K-MDP algorithm

Our second algorithm is based on a transitive approximate abstraction function $\phi_{Q_d^*}$, introduced by Abel et al. (2018).

Approximate transitive $\phi_{Q_d^*}$ function abstraction For any original states i, j, an approximate transitive state abstraction function satisfies:

$$
\phi_{Q_d^*}(i) = \phi_{Q_d^*}(j) \implies \forall a \left\lceil \frac{Q^*(i, a)}{d} \right\rceil = \left\lceil \frac{Q^*(j, a)}{d} \right\rceil \tag{4}
$$

for $0 < d \leq$ VMAX, where VMAX represents the maximum optimal value. A transitive state abstraction function $\phi_{Q_d^*}$, as reported by Abel et al. (2018), has a value loss that scales in accordance with d:

$$
\forall s \in S, V^{\pi^*}(s) - V^{\pi_K^*}_{\phi_{Q_d^*}}(s) \leq \frac{2dR_{max}}{(1 - \gamma)^2}.
$$

That we reformulate as,

$$
\max_{s \in S} V^{\pi^*}(s) - V^{\pi_K^*}_{\phi_{Q_d^*}}(s) \leq \frac{2dR_{max}}{(1 - \gamma)^2}. \tag{5}
$$

Unlike state abstraction function $\phi_{Q_\epsilon^*}$, transitive predicates offer a unique minimal state abstraction for which all state pairs that satisfy the predicate belong to the same cluster. In the case of $\phi_{Q_d^*}$, a set of states belong to the same cluster if they belong to the same bin as defined by Eq. 4.

The $\phi_{Q_d^*}$ K-MDP algorithm explained We now propose the $\phi_{Q_d^*}$ K-MDP algorithm (Alg. 3) to minimize the value of d given a p_{target} parameter and the abstraction function $\phi_{Q_d^*}$. Alg. 3 takes as argument an MDP model M, a parameter $K \geq 1$ denoting a maximum number of states for the K-MDP, and a precision value p_{target}. Alg. 3 performs a binary search on d by setting the first upper bound d^+ to VMAX and lower bound d^- to 0. For all states $s \in S$ and

Algorithm 3 $\phi_{Q_d^*}$ K-MDP

Require: $M, K \geq 1, p_{target}, Q^*$
1: $d^+ = \text{VMAX}, d^- = 0$
2: **repeat**
3: $p = d^+ - d^-$
4: $d = d^- + \frac{d^+ - d^-}{2}$
5: **for** $s \in S$ **do**
6: **for** $a \in A$ **do**
7: $bindings \leftarrow \lceil Q^*(s,a)/d \rceil$
8: $S_K \leftarrow$ *unique*$(bindings)$
9: **if** $|S_K| \leq K$ **then**
10: $d^+ = d$
11: **else**
12: $d^- = d$
13: **until** $p < p_{target}$
14: $M_K \leftarrow$ *BUILD-K-MDP*(M, S_K, ϕ)
15: **return** M_K

actions $a \in A$, we compute the *ceil* values given parameter d (Eq. (4), line 7). $bindings$ contains the *ceil* values for all states and actions. Function *unique* returns an abstracted state space S_K by grouping states that belong to the same bin (line 8). Finally, the upper bound d^+ or lower bound d^- are updated. The algorithm continues until the precision criterion is reached. Alg. 3 has time complexity $O(|S| \log(\frac{\text{VMAX}}{p_{target}}))$ due to the binary search and the *unique* procedure, which is linear in $|S|$.

Proposition 2. *Alg. 3 solves the K-MDP problem within precision p_{target} in a finite number of iterations. The optimal value function derived from the solution M_K, $V^{\pi_K^*}_{\phi_{Q_d^*}}$, has sub-optimality bounded polynomially in d.*

Proof. This follows from the binary search and Eq. (5). □

The $\phi_{a_d^*}$ K-MDP algorithm

Abel et al. (2018) were motivated by a reinforcement learning problem and did not focus on state approximation function based on optimal value functions. However, they mentioned the opportunity to apply the ceiling discretization approach to transform the exact transitive function ϕ_a^* into an approximate transitive function. This is what we did, denoting this state abstraction function $\phi_{a_d^*}$.

Approximate transitive $\phi_{a_d^*}$ function abstraction We first introduce the exact state abstraction function ϕ_{a^*}. For any original states i, j, the exact state abstraction function ϕ_{a^*} satisfies:

$$\phi_{a^*}(i) = \phi_{a^*}(j) \implies a_i^* = a_j^* \wedge V^*(i) = V^*(j).$$

ϕ_{a^*} preserves the optimal actions and their values (Li, Walsh, and Littman 2006). Now applying the ceiling function, the approximate transitive state abstraction function $\phi_{a_d^*}$ satisfies:

$$\phi_{a_d^*}(i) = \phi_{a_d^*}(j) \implies a_i^* = a_j^* \wedge \left\lceil \frac{V^*(i)}{d} \right\rceil = \left\lceil \frac{V^*(j)}{d} \right\rceil \quad (6)$$

for $0 < d \leq \text{VMAX}$ and $a_i^*, a_j^* \in A$, where a_i^* and a_j^* are the optimal actions to implement in states i and j respectively. In other words, two given states i and j can be aggregated if they belong to the same bin and their optimal actions are the same. One can show that the value loss result in Eq. 5 from $\phi_{Q_d^*}$ extends to $\phi_{a_d^*}$.

Intuitively, using $\phi_{a_d^*}$ as state abstraction function is more likely to minimize the gap of Eq. (1) than $\phi_{Q_d^*}$ because it preserves the optimal actions. However, this advantage comes at a cost of not finding a feasible state abstraction when the number of unique optimal actions in the original optimal MDP policy π^* is greater than K.

One can show that the value loss result from $\phi_{Q_d^*}$ extends to $\phi_{a_d^*}$. A transitive state abstraction function $\phi_{a_d^*}$ has a value loss that scales in accordance with d:

$$\forall s \in S, V^{\pi^*}(s) - V^{\pi_K^*}_{\phi_{a_d^*}}(s) \leq \frac{2dR_{max}}{(1-\gamma)^2}.$$

Alternatively,

$$\max_{s \in S} V^{\pi^*}(s) - V^{\pi_K^*}_{\phi_{a_d^*}}(s) \leq \frac{2dR_{max}}{(1-\gamma)^2}. \quad (7)$$

The $\phi_{a_d^*}$ K-MDP algorithm explained We can naturally propose the $\phi_{a_d^*}$ K-MDP algorithm (Alg. 4) to find the minimum value of d that returns a set of abstracted states S_K, where $|S_K| \leq K$, given a precision p_{target} parameter and the abstraction function $\phi_{a_d^*}$. Alg. 4 is similar to Alg. 3 except that lines 5:7 are replaced by lines 1:3.

Algorithm 4 $\phi_{a_d^*}$ K-MDP

M
1: **for** $s \in S$ **do**
2: $a_s^* \leftarrow \pi^*(s)$
3: $bindings \leftarrow [\lceil \frac{V^*(s)}{d} \rceil, a_s^*]$

In this case, $bindings$ is a data structure that stores the *ceil* value of $\frac{V^*(s)}{d}$ and the optimal action a_s^*. Alg. 4 has time complexity $O(|S| \log(\frac{\text{VMAX}}{p_{target}}))$ due to the binary search and the *unique* procedure.

Proposition 3. *Alg. 4 solves the K-MDP problem within precision p_{target} in a finite number of iterations if K is greater than or equal to the number of unique actions of the original optimal MDP policy π^*. The optimal value function derived from the solution M_K, $V^{\pi_K^*}_{\phi_{a_d^*}}$, has sub-optimality bounded polynomially in d.*

Proof. This proposition is a direct consequence of the binary search algorithm, Predicate 6 and Eq. 7. □

k-means++ K-MDP algorithm

The k-means algorithm seeks to minimize the average squared distance between points in the same cluster. Given an integer k and a set of n data points $X \in \Re^d$, the k-means problem seeks to choose k centers $\mathcal{C}$ so as to minimize:

$$\sum_{x \in X} \min_{c \in \mathcal{C}} ||x - c||^2.$$

Solving the k-means problem is NP-hard. Although efficient in practice, k-means offers no accuracy guarantees. k-means++ augments k-means with a randomized seeding technique. We propose to use k-means++ with the optimal Q function and norm $L1$ to define the state space S_K:

$$\sum_{s \in S} \min_{s_K \in S_K} ||Q^*(s, \cdot) - Q^*(s_K, \cdot)||^2.$$

By using k-means++ clustering, we hope for a good clustering that will yield to a small error since we have lost the performance guarantee discussed previously.

Algorithm 5 k-means++ K-MDP

Require: $M, Q, K \geq 1$
1: $S_K, \phi \leftarrow k$-means++(Q, K)
2: $M_K \leftarrow$ *BUILD-K-MDP*(M, S_K, ϕ)
3: **return** M_K

Given an MDP, its *Q-values* and a target number of reduced states K, Alg. 5 performs a k-means++ clustering where *Q-values* are the data points and the results are the cluster indices for each one of the original states. k-means++ is $O(\log k)$-competitive with the optimal clustering (Arthur and Vassilvitskii 2007). k-means is based on Lloyd's algorithm (Lloyd 1982) and has time complexity $O(nkdi)$, where k is the number of clusters, n is the number of data points, d the number of dimensions and i the number of iterations required to converge (Hartigan and Wong 1979).

$\phi_{Q^*_\epsilon}$ Greedy K-MDP algorithm

Our last computational approach is a straight forward algorithm that performs a binary search on ϵ and greedily aggregates states following a $\phi_{Q^*_\epsilon}$ state abstraction function. The order in which states are considered is randomized.

Experimental Results

We ran our experiments on an Intel Core i7-8650U with a 1.90 GHz clock and a memory of 16 GB for all algorithms. All experiments were conducted using the *MDP-Toolbox* (Chadès et al. 2014), MATLAB (R2019b) and CPLEX (12.5). We aimed to cover a range of problems, so our applications have reasonable branching factors while random generated MDPs have large branching factors. For simplicity, we also assumed that, for any given abstract state s_K, if the number of original states aggregated to s_K is $|\phi^{-1}(s_K)|$, then the weight of each $s \in \phi^{-1}(s_K)$ is uniformly distributed: $\omega(s) = 1/|\phi^{-1}(s_K)|$. The resulting abstract K-MDP is not Markovian. When aggregating states, we lose full observability and the problem could be modelled as a POMDP: a transition probability from s to s' (abstract) given action a depends on the history of abstract states and actions, and on the initial probability distribution over original states. Given i) an original MDP, ii) an observation function ϕ mapping states to abstract states and iii) an initial probability distribution over states, we can solve the induced abstract MDP as a POMDP - where abstract states are observations. Here, we are ignoring this and heuristically approximating the problem as an MDP. We then evaluate those abstract policies on the original MDP to assess performance of our algorithms. In the future, we plan to explore how to obtain more accurate transition probabilities.

Randomly Generated MDPs

We generated 100 instances of random MDPs for $|S| = 1000$ and $|A| \in \{4, 50\}$, $|S| \in \{2500, 5000\}$ and $|A| = 4$, and a branching factor of $|S|$ ($|S|$ states could be reached for each pair s, a). Tab. 1 shows the average per-instance gap (%) and standard deviation for each algorithm. For $\phi_{Q^*_d}$, $\phi_{a^*_d}$ and $\phi_{Q^*_\epsilon}$ Greedy K-MDP algorithms, we set up a precision target of 0.0001 (p_{target}). $\phi_{a^*_d} K$-MDP outperformed all other algorithms with no value loss. The other algorithms had larger errors with k-means++ K-MDP being second best. $\phi_{Q^*_\epsilon}$ Greedy K-MDP algorithm performed worse.

$\phi_{Q^*_\epsilon}$ K-MDP-ILP evaluation It was not possible to run experiments using $\phi_{Q^*_\epsilon} K$-MDP-ILP for problems with $|S| > 500$ due to memory requirements. We report separately the performance of $\phi_{Q^*_\epsilon} K$-MDP-ILP for a precision target of 0.02 on randomly generated problems with up to 500 states and 10 actions on 10 instances, and compare the loss of performance against other algorithms (Tab. 2). $\phi_{Q^*_\epsilon} K$-MDP-ILP under-performed for $|S|/2$ and performed similarly to $\phi_{Q^*_\epsilon} K$-MDP-ILP, $\phi_{Q^*_d} K$-MDP and $\phi_{Q^*_\epsilon}$ Greedy K-MDP for other K values. Algorithm $\phi_{a^*_d} K$-MDP systematically minimized the performance loss for these instances.

Computational Sustainability Case Studies

Recovering two endangered species The sea otter (*Enhydra lutris kenyoni*) and its preferred prey, northern abalone (*Haliotis kamtschatkana*) are both listed as endangered in British Columbia, Canada. This classic conservation problem was described by Chades, Curtis, and Martin (2012). The objective is to maximize the abundance of both species over time. The original MDP has 819 states representing the population of sea otters and the density of northern abalone. Managers can choose between 4 actions: Introducing sea otters (I), antipoaching measures (AP), control sea otters (C) and one half antipoaching and one half control sea otters (H). Antipoaching reduces the illegal harvesting of abalone by 50%, control sea otters remove the number of otters above 60% of their carrying capacity. Half antipoaching and half control sea otters takes both actions simultaneously but at half the level of effectiveness. Fig. 1 shows the performance of k-means++ K-MDP, $\phi_{Q^*_d}$ K-MDP, $\phi_{a^*_d}$ K-MDP and $\phi_{Q^*_\epsilon}$ Greedy K-MDP. All algorithms except $\phi_{Q^*_\epsilon}$ Greedy K-MDP performed well with $\phi_{a^*_d}$ K-MDP providing the small-

	k-means++ K-MDP					$\phi_{Q_d^*}$ K-MDP					$\phi_{a_d^*}$ K-MDP					$\phi_{Q_\epsilon^*}$ Greedy K-MDP				
Instance	$\|S\|/2$	$\|S\|/8$	$\|S\|/15$	$\|S\|/30$	$\|S\|/100$	$\|S\|/2$	$\|S\|/8$	$\|S\|/15$	$\|S\|/30$	$\|S\|/100$	$\|S\|/2$	$\|S\|/8$	$\|S\|/15$	$\|S\|/30$	$\|S\|/100$	$\|S\|/2$	$\|S\|/8$	$\|S\|/15$	$\|S\|/30$	$\|S\|/100$
$\|S\| = 1000$ $\|A\| = 4$	0.0 ±0.0	0.1 ±0.0	0.2 ±0.0	0.3 ±0.0	0.6 ±0.0	0.1 ±0.0	0.4 ±0.0	0.6 ±0.1	1.3 ±0.1	1.7 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.8 ±0.0	1.7 ±0.0	1.9 ±0.0	2.1 ±0.0	2.2 ±0.0
$\|S\| = 1000$ $\|A\| = 50$	1.3 ±0.0	2.4 ±0.0	2.8 ±0.0	3.0 ±0.0	2.8 ±0.0	0.9 ±0.0	2.8 ±0.0	3.6 ±0.0	4.3 ±0.0	4.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	1.6 ±0.0	3.2 ±0.0	3.6 ±0.0	4.0 ±0.0	3.8 ±.0
$\|S\| = 2500$ $\|A\| = 4$	0.0 ±0.0	0.0 ±0.0	0.1 ±0.0	0.2 ±0.0	0.2 ±0.0	0.0 ±0.0	0.2 ±0.0	0.3 ±0.0	0.4 ±0.0	0.7 ±0.1	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.5 ±0.0	1.0 ±0.0	1.2 ±0.0	1.3 ±0.0	1.4 ±0.0
$\|S\| = 5000$ $\|A\| = 4$	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.2 ±0.0	0.0 ±0.0	0.0 ±0.0	0.1 ±0.0	0.2 ±0.0	0.5 ±0.1	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.4 ±0.0	0.8 ±0.0	0.9 ±0.0	0.9 ±0.0	1.0 ±0.0

Table 1: Mean gap (%) and standard deviation of our K-MDP algorithms using randomly generated MDPs.

	$\phi_{Q_\epsilon^*}$ K$-MDP-ILP			k-means++ K-MDP			$\phi_{Q_d^*}$ K-MDP			$\phi_{a_d^*}$ K-MDP			$\phi_{Q_\epsilon^*}$ Greedy K-MDP		
Instance	$\|S\|/2$	$\|S\|/6$	$\|S\|/10$	$\|S\|/2$	$\|S\|/6$	$\|S\|/10$	$\|S\|/2$	$\|S\|/6$	$\|S\|/10$	$\|S\|/2$	$\|S\|/6$	$\|S\|/10$	$\|S\|/2$	$\|S\|/6$	$\|S\|/10$
$\|S\|$=100,$\|A\|$=10	6.5±0.3	7.9±0.2	8.5±0.1	1.4±0.1	3.3±0.5	4.1±0.5	4.0±0.1	7.7±0.3	8.0±0.5	0.0±0.0	0.0±0.0	0.0±0.0	3.2±0.3	6.3±0.3	7.4±0.3
$\|S\|$=200,$\|A\|$=10	6.4±0.2	4.6±0.2	5.6±0.8	0.9±0.0	2.0±0.0	2.5±0.1	2.0±0.1	5.9±0.1	6.3±0.3	0.0±0.0	0.0±0.0	0.0±0.0	2.3±0.1	4.5±0.1	5.1±0.3
$\|S\|$=300,$\|A\|$=10	6.3±0.2	3.9±0.07	4.6±0.2	0.7±0.0	1.5±0.2	1.8±0.1	3.0±0.1	3.7±0.1	4.4±0.1	0.0±0.0	0.0±0.0	0.0±0.0	1.9±0.0	3.7±0.1	4.2±0.0
$\|S\|$=400,$\|A\|$=10	5.5±0.0	3.4±0.1	3.9±0.1	0.6±0.0	1.3±0.1	1.5±0.1	2.9±0.1	3.6±0.1	3.6±0.1	0.0±0.0	0.0±0.0	0.0±0.0	1.8±0.1	3.6±0.1	3.8±0.2
$\|S\|$=500,$\|A\|$=10	4.5±0.0	3.0±0.0	3.4±0.1	0.5±0.0	1.0±0.0	1.4±0.0	2.6±0.1	3.3±0.1	3.3±0.1	0.0±0.0	0.0±0.0	0.0±0.0	1.6±0.1	3.0±0.1	3.3±0.1

Table 2: Mean gap (%) and standard deviation percentages of our K-MDP, including $\phi_{Q_\epsilon^*}$ K-MDP-ILP and using randomly generated MDPs

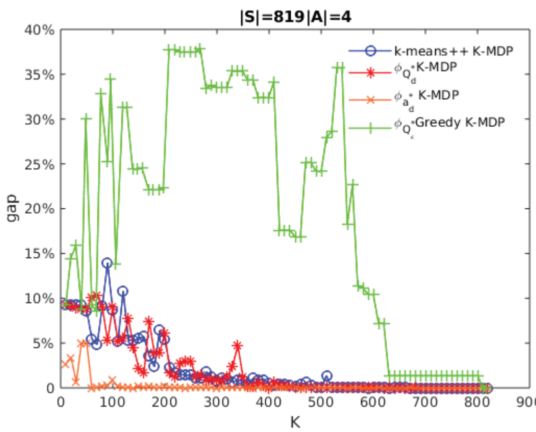


Figure 1: Performance of K-MDP algorithms on the sea otter and northern abalone conservation problem.

est gap consistently. We investigated the interpretability of both the model and the optimal K-MDP policy generated by $\phi_{a_d^*}K$-MDP (Fig. 2). The reported gap for K=10 was less than 2.8%. The K-MDP policy graph was consistent with the original optimal policy. However, the original optimal policy would have required 819 nodes with at most 2^{819} edges, instead of 10 nodes with at most 2^{10} edges. Reducing the state space to a small K clearly results in a more interpretable policy.

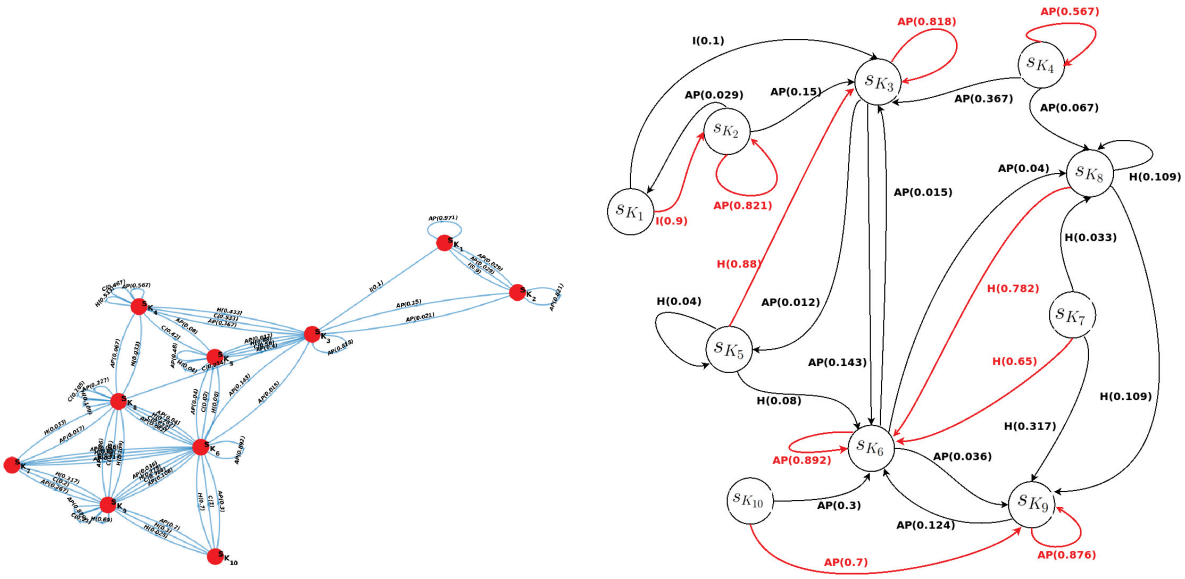
Managing an ecological network of invasive species *Aedes albopictus* is an invasive mosquito species that persists in the Torres Strait Islands (North of Australia). In this problem introduced by Péron et al. (2017), decision makers require guidance to best delay the time of infestation of mainland Australia. The problem is modeled as a dynamic Susceptible-Infected-Susceptible (SIS) network with yearly probability of re-infestation. Every year, managers can implement simultaneous actions of different intensity and duration (light and strong) at two locations at most. Locations (islands in this case) can be either *infested* or *susceptible* to be infested. Péron et al. (2017) modeled the problem with up to 17 islands—however the problem is intractable for a standard computer with more than 13 islands. For the purpose of our study, we generated a simplified problem with 3 islands (6097 states and 17 management actions) and evaluated our K-MDP algorithms for values of K ranging from 4878 to 10 (Tab. 3). For $K > 10$, results provided by all algorithms were attractive with a maximum gap of 3.8% for k-means++ K-MDP and a maximum gap of 0.7% for $\phi_{a_d^*}K$-MDP. However, for $K = 10$, $\phi_{a_d^*}K$-MDP was not able to find a feasible state abstraction while the other algorithms underperformed with a gap $> 97.8\%$.

Conclusion

Modelling MDP problems remain an art. Our approach "solving K-MDPs" aims at increasing trust and uptake of MDPs in human operated systems by providing easier to interpret models and solutions.

Our algorithms have sub-optimality bounded polynomially in a discretization parameter (ϵ or d). Using a non-transitive approximate state abstraction function led us to develop a pure integer linear program ILP_{KMDP} to solve a node clique cover decision problem. While mathematically elegant, ILP_{KMDP} was untractable for all but small problems. We have then employed transitive approximate state abstraction functions to solve K-MDPs in time complexity $O(n \log m)$. We have also proposed to use a clustering technique k-means++ and a binary search greedy approach. Experimental results showed that, while all algorithms performed reasonably well, $\phi_{a_d^*}K$-MDP was clearly the best performer when it found a solution.

Immediate future work will focus on testing the interpretability of proposed K-MDPs solutions with domain experts and behavioural scientists. Our current algorithms require solving the original MDPs in the first place. This is because we focused on providing interpretable policies of existing MDPs, rather than solving large state space MDPs. Future research could exploit the structure of large MDPs and reduce their size to tractable MDPs before using our approaches.



(a) Sea Otter and Northern Abalone K-MDP model.

(b) Sea Otter and Northern Abalone K-MDP policy graph.

Figure 2: K-MDP model (a) and policy graph (b) for $K = 10$ for the Sea Otter and Northern Abalone problem generated by $\phi_{a_d^*} K$-MDP. In the K-MDP model, red nodes represent the abstracted states and blue edges are transitions T_K. In the policy graph, red edges represent the most likely transition resulting of applying the optimal policy. Edges are labeled by the action name and its probability to transition from one state s_K to another state s'_K. The starting state is S_{K_1}: low abalone density and absence of sea otters. Note that because S_{K_4} and $S_{K_{10}}$ are unreachable from S_{K_1}, the policy graph could be further simplified.

	k-means++ K-MDP			$\phi_{Q_d^*}$ K-MDP			$\phi_{a_d^*}$ K-MDP			$\phi_{Q_\epsilon^*}$ Greedy K-MDP		
$\lvert S_K \rvert$	**G(%)**	**B(sec.)**	**T(s.)**	**G(%)**	**B(sec.)**	**T(sec.)**	**G(%)**	**B(sec.)**	**T(sec.)**	**G(%)**	**B(sec.)**	**T(sec.)**
4878	0.0	148.42	1909.17	0.0	137.44	1923.72	0.0	142.21	1952.59	0.00	828.32	1888.10
2032	0.0	39.49	242.79	0.0	37.02	244.5	0.0	38.42	244.19	2.6	796.42	259.17
1219	0.1	21.7	124.65	0.1	20.6	126.62	0.0	21.65	127.51	3.6	752.14	127.69
610	0.2	12.82	82.39	0.4	12.61	83.88	0.0	13.4	85.22	2.6	748.59	81.71
102	3.8	7.55	82.797	2.4	7.60	83.92	0.7	7.478	83.56	3.1	932.12	73.41
10	97.8	7.64	43.155	97.8	8.16	9.89	-	-	-	-	>1000	-
MDP	**Solving MDP(sec.)**											
6097	**2925.34**											

Table 3: Performance results on the *Aedes Albopictus* problem. Columns report the gap (G), the time required to build (B) the K-MDP with K states and the time to solve (T) the K-MDP using value iteration. Last row shows $|S|$ and the time to compute the original optimal policy using value iteration.

References

Abel, D.; Arumugam, D.; Lehnert, L.; and Littman, M. L. 2017. Toward good abstractions for lifelong learning. In *NIPS workshop on Hierarchical Reinforcement Learning*.

Abel, D.; Arumugam, D.; Lehnert, L.; and Littman, M. 2018. State abstractions for lifelong reinforcement learning. In *ICML*.

Abel, D.; Hershkowitz, D.; and Littman, M. 2016. Near optimal behavior via approximate state abstraction. In *ICML*, volume 48 of *PMLR*, 2915–2923.

Andre, D., and Russell, S. J. 2002. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, 119–125.

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proc. of the 18th annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035.

Bellman, R. 1958. *Dynamic programming*. Princeton University Press, John Wiley & Sons.

Bertram, J., and Wei, P. 2018. Explainable deterministic MDPs. *arXiv preprint arXiv:1806.03492*.

Brigham, R. C., and Dutton, R. D. 1983. On clique covers and independence numbers of graphs. *Discrete Mathematics* 44(2):139–144.

Chadès, I.; Chapron, G.; Cros, M.-J.; Garcia, F.; and Sabbadin, R. 2014. MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography* 37(9):916–920.

Chades, I.; Curtis, J. M. R.; and Martin, T. G. 2012. Setting realistic recovery targets for two interacting endangered species, sea otter and northern abalone. *Conservation Biology* 26(6):1016–1025.

Chakraborti, T.; Sreedharan, S.; Grover, S.; and Kambhampati, S. 2019. Plan explanations as model reconciliation. In *HRI*. IEEE.

Dean, T., and Givan, R. 1997. Model minimization in Markov decision processes. In *AAAI/IAAI*, 106–111.

Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89(1-2):219–283.

Dietterich, T. G. 2000. State abstraction in maxq hierarchical reinforcement learning. In *NIPS*, 994–1000.

Dujardin, Y.; Dietterich, T.; and Chadès, I. 2015. α-min: A compact approximate solver for finite-horizon POMDPs. In *IJCAI*, 2582–2588.

Dujardin, Y.; Dietterich, T.; and Chadès, I. 2017. Three new algorithms to solve N-POMDPs. In *AAAI*, 4495–4501.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2019. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51(5):93.

Hartigan, J. A., and Wong, M. A. 1979. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Stat. Soc. Series C (Applied Statistics)* 28(1):100–108.

Jong, N. K., and Stone, P. 2005. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, 752–757.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer. 85–103.

Krarup, B.; Cashmore, M.; Magazzeni, D.; and Miller, T. 2019. Model-based contrastive explanations for explainable planning. In *ICAPS 2019 Workshop on Explainable AI Planning (XAIP)*. AAAI Press.

Lakkaraju, H., and Rudin, C. 2017. Learning cost-effective and interpretable treatment regimes. In *Artificial Intelligence and Statistics*, 166–175.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for MDPs. In *ISAIM*.

Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137.

Marescot, L.; Chapron, G.; Chadès, I.; Fackler, P. L.; Duchamp, C.; Marboutin, E.; and Gimenez, O. 2013. Complex decisions made simple: a primer on stochastic dynamic programming. *Methods in Ecology and Evolution* 4(9):872–884.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of operations research* 12(3):441–450.

Péron, M.; Jansen, C. C.; Mantyka-Pringle, C.; Nicol, S.; Schellhorn, N. A.; Becker, K. H.; and Chadès, I. 2017. Selecting simultaneous actions of different durations to optimally manage an ecological network. *Methods in Ecology and Evolution* 8(10):1332–1341.

Petrik, M., and Luss, R. 2016. Interpretable policies for dynamic product recommendations. In *UAI*.

Pineau, J.; Gordon, G.; and Thrun, S. 2002. Policy-contingent abstraction for robust robot control. In *UAI*.

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

Singh, S. P.; Jaakkola, T.; and Jordan, M. I. 1995. Reinforcement learning with soft state aggregation. In *NIPS*, 361–368.

Vellido, A.; Martín-Guerrero, J. D.; and Lisboa, P. J. 2012. Making machine learning models interpretable. In *ESANN*, volume 12, 163–172.

Vianna, L. G.; Sanner, S.; and De Barros, L. N. 2013. Bounded approximate symbolic dynamic programming for hybrid MDPs. *arXiv preprint arXiv:1309.6871*.