α-min: A Compact Approximate Solver For Finite-Horizon POMDPs

Yann Dujardin
CSIRO
yann.dujardin@csiro.au

Tom Dietterich
School of EECS
Oregon State University
tgd@oregonstate.edu

Iadine Chadès
CSIRO
iadine.chades@csiro.au

Abstract

In many POMDP applications in computational sustainability, it is important that the computed policy have a simple description, so that it can be easily interpreted by stakeholders and decision makers. One measure of simplicity for POMDP value functions is the number of α-vectors required to represent the value function. Existing POMDP methods seek to optimize the accuracy of the value function, which can require a very large number of α-vectors. This paper studies methods that allow the user to explore the tradeoff between the accuracy of the value function and the number of α-vectors. Building on previous point-based POMDP solvers, this paper introduces a new algorithm (α-min) that formulates a Mixed Integer Linear Program (MILP) to calculate approximate solutions for finite-horizon POMDP problems with limited numbers of α-vectors. At each time-step, α-min calculates α-vectors to greedily minimize the gap between current upper and lower bounds of the value function. In doing so, good upper and lower bounds are quickly reached allowing a good approximation of the problem with *few α-vectors*. Experimental results show that α-min provides good approximate solutions given a fixed number of α-vectors on small benchmark problems, on a larger randomly generated problem, as well as on a computational sustainability problem to best manage the endangered Sumatran tiger.

1 Introduction

Most Partially Observable Markov Decision Process (POMDP) algorithms focus on providing near optimal solutions to infinite-horizon POMDP problems. Near-optimal performance comes at the cost of providing solutions requiring many alpha vectors even when solving small size POMDP problems [Poupart *et al.*, 2011]. In applied fields such as conservation biology, POMDP solutions are often too complex to be analyzed and communicated to managers to be implemented [Tulloch *et al.*, 2015]. Indeed, deriving simple management rules has proven difficult when the POMDP solution has a large number of alpha vectors [Nicol and Chadès, 2012]. As a result, the solutions must be explored by simulations in order to derive general rules of thumb [Chadès *et al.*, 2011]. In the best case, these rules are tested via simulations and the loss of performance is reported. This practice is time consuming and does not offer performance guarantees. There is a need to provide an alternative for POMDP users who require simple approximate solutions. To tackle this issue, we develop α-min, a finite-horizon POMDP solver that calculates a good policy given a limit on the number of α-vectors.

Section 2 provides an overview of POMDPs. Section 3 introduces the principles of our approach that relies on calculating α-vectors that minimize the gap between a tight upper bound and a current lower bound approximating the optimal value function. In section 4, we provide a MILP formulation of our approach and present the properties of a first algorithm ϵ-min that finds approximate solutions given a fixed maximum gap. From ϵ-min, we then derive α-min that finds good approximate solutions given a fixed number of α-vectors. Section 5 assesses the performance of α-min on four small benchmark problems, on a larger randomly generated problem, and on a novel computational sustainability problem that seeks to best allocate resources to protect a meta-population of threatened Sumatran tigers. Finally, we discuss the results and future works in Section 6.

Table 3 contains the main notations of this paper. Proofs and proof sketches are available as supplementary material[1].

2 POMDP Overview

POMDPs are a convenient model for solving sequential decision-making optimization problems when the decision-maker does not have complete information about the current state of the system [Sigaud and Buffet, 2013]. Formally, a discrete finite-horizon POMDP is specified as a tuple $\{S, A, O, \tau_a, \Omega_a, R, H\}$, where

- $H = \{0, ..., T-1\}$, $T \in \mathbb{N}$, is the time horizon. Elements of H are called time-steps and T is the number of time-steps.
- $S, \forall t \in H,\ s_t \in S$ is the state of the system at t.
- $A, \forall t \in H,\ a_t \in A$ is the taken action at t.
- $O, \forall t \in H,\ z_t \in O$ is the observation at t.

[1] https://sites.google.com/site/ijcaialphamin/home

- τ_a is the transition matrix for action a. Elements are $\tau_a\left(s_t, s_{t+1}\right)$.
- Ω_a is the observation matrix for action a. Elements are $\Omega_a(s_{t+1}, z_{t+1})$.
- R is the reward matrix. Elements are $R\left(a_t, s_t\right)$.

For sake of clarity, we define the following notation:

- For action $a \in A$ and for observation $z \in O$, let $M_{a,z}$ be the matrix of dimension $S \times S$ such that $M_{a,z}(s_{t+1}, s_t) = \Omega_a(z, s_{t+1})\tau_a(s_{t+1}, s_t)$.
- For every $a \in A$, the vector $\boldsymbol{r_a} = \boldsymbol{R(a,.)}$ corresponds to the row of the matrix R corresponding to the action a.

The optimal decision at time t may depend on the complete history of past actions and observations. Because it is neither practical nor tractable to use the history of the action-observation trajectory to compute an optimal solution, belief states (also called beliefs), i.e. probability distributions over states, are used to summarize and overcome the difficulties of imperfect detection [Åström, 1965]. A POMDP can be cast into a fully observable Markov decision process defined over the continuous belief state space, B.

Solving exactly a finite-horizon POMDP means finding an optimal policy $\Pi_0 = \cup_{t\in H}\pi_t$, where $\pi_t : B \to A$ maps belief states at time t to actions. Π_0 maximizes the expected sum of rewards $E[\sum_{t\in H} \boldsymbol{r_{a_t}} \cdot \boldsymbol{b_t}]$ over the time horizon H ($\cdot$ denotes the scalar product). For each time-step t, for a given belief state $\boldsymbol{b_t}$ and a given policy $\Pi_t = \cup_{t'\in\{t,\ldots,T-1\}}\pi_{t'}$ the expected sum $E[\sum_{t'\in\{t,\ldots,T-1\}} \boldsymbol{r_{a_t}} \cdot \boldsymbol{b_t}]$ is also referred to as the value function $V_{t,\Pi_t}(\boldsymbol{b_t})$. A value function allows us to rank strategies by assigning a real value to each belief $\boldsymbol{b_t}$. An optimal policy Π_t is a policy such that, $\forall\ \boldsymbol{b_t} \in B$, $\forall\ \Pi'_t$, $V_{t,\Pi_t}(\boldsymbol{b_t}) \geq V_{t,\Pi'_t}(\boldsymbol{b_t})$. Several strategies can be optimal and share the same optimal value function V_t, which can be computed using Bellman's principle of optimality [Bellman, 1957]: $\forall \boldsymbol{b_t} \in B$,

$$V_t(\boldsymbol{b_t}) = \max_{a_t\in A}\{\boldsymbol{r_{a_t}} \cdot \boldsymbol{b_t} + \sum_{z_t\in O} p(z_{t+1}|a_t, \boldsymbol{b_t})V_{t+1}(\boldsymbol{b_{t+1}})\} \tag{1}$$

where the belief $\boldsymbol{b_{t+1}}$ can be computed as follows : $\forall s_{t+1} \in S$,

$$b_{t+1}(s_{t+1}) = \frac{\Omega_a(s_{t+1}, z_{t+1})\sum_{s_t\in S}\tau_a(s_t, s_{t+1})b_t(s_t)}{\sum_{s_t, s'_{t+1}\in S}\Omega_a(s'_{t+1}, z_{t+1})\tau_a(s_t, s'_{t+1})b_t(s_t)} \tag{2}$$

Equation 1 can be rewritten $V_t = BL(V_{t+1})$ where BL is the Bellman operator [Shani *et al.*, 2012], sometimes also called backup operator [Pineau *et al.*, 2006]. While various algorithms from the operations research and artificial intelligence literature have been developed over the past years, exact resolution of POMDPs is intractable: finite-horizon POMDPs are PSPACE-complete [Papadimitriou and Tsitsiklis, 1987] and infinite-horizon POMDPs are undecidable [Madani *et al.*, 2003].

3 α-min principle

Equation 1 can be solved by directly manipulating α-vectors [Smallwood and Sondik, 1973]. For every $t \in H$, there exists a finite set Γ_t of vectors of dimension $|S|$ (the so-called α-vectors) which define entirely V_t such as $|\Gamma_t|$ is minimal and:

$$\forall t \in H, \forall \boldsymbol{b_t} \in B, V_t\left(\boldsymbol{b_t}\right) = \max_{\boldsymbol{\alpha_t}\in\Gamma_t} \boldsymbol{\alpha_t} \cdot \boldsymbol{b_t} \tag{3}$$

This formulation is equivalent to equation 1. Given that $\Gamma_{T-1} = \{\boldsymbol{r_a}|\ a \in A,\ \boldsymbol{r_a}$ is not dominated$\}$, one can in theory build a set of α-vectors defining the value function V_t from Γ_{t+1} at any time-step $t \in H' = \{0, ..., T-2\}$, because every $\boldsymbol{\alpha_t} \in \Gamma_t$ can be written:

$$\boldsymbol{\alpha_t} = [\boldsymbol{r_{a_t}} + \sum_{z_{t+1}\in O}(\boldsymbol{\alpha_{t+1}^{a_t,z_{t+1}}})^T M_{a_t,z_{t+1}}]^T \tag{4}$$

where $a_t \in A$ and $\boldsymbol{\alpha_{t+1}^{a_t,z_{t+1}}}, z_{t+1} \in O$ are elements of Γ_{t+1}.

The set of α-vectors $P(\Gamma_{t+1})$ given by Equation 4 is such that $\Gamma_t \subseteq P(\Gamma_{t+1})$ (α-vectors of $P(\Gamma_{t+1})$ *potentially* belong to Γ_t). In the case of exact resolution, for a given Γ_{t+1}, a natural way to compute Γ_t is first to compute $P(\Gamma_{t+1})$ entirely, and then prune the dominated vectors that are not useful for representing the value function. In practice, this approach is computationally expensive [Sigaud and Buffet, 2013, sections 7.3 and 7.4].

Point-based algorithms are recent approximate approaches to solve POMDPs [Shani *et al.*, 2012]. Value functions are updated according to a subset of beliefs $\tilde{B}_t \subseteq B$ sampled to be as relevant as possible to get good approximations of Γ_t at each time-step. Every alpha vector $\boldsymbol{\alpha_t^{b_t}}$ of the current approximation $\tilde{\Gamma}_t \subseteq \Gamma_t$, corresponding to the belief $\boldsymbol{b_t} \in \tilde{B}_t$, is generated as follows:

$$\boldsymbol{\alpha_t^{b_t}} = \operatorname*{arg\,max}_{\boldsymbol{\alpha_t}\in P(\tilde{\Gamma}_{t+1})} \boldsymbol{\alpha_t} \cdot \boldsymbol{b_t} \tag{5}$$

where $\tilde{\Gamma}_{t+1}$ is an approximation of Γ_{t+1}.

Definition 1. *We call $\overline{\boldsymbol{\alpha}}$ the vector-function such that for every $\boldsymbol{b_t} \in B$, $\overline{\boldsymbol{\alpha}}(\boldsymbol{b_t}) = \arg\max_{\boldsymbol{\alpha_t}\in P(\tilde{\Gamma}_{t+1})} \boldsymbol{\alpha_t} \cdot \boldsymbol{b_t}$. Note that for every $\boldsymbol{b_t} \in B$, we have $\overline{\boldsymbol{\alpha}}(\boldsymbol{b_t}) \cdot \boldsymbol{b_t} = BL(\tilde{V}_{t+1})(\boldsymbol{b_t})$.*

Using the point-based approach on a finite-horizon, one can build a lower approximation of the optimal value function at each time-step. We start by setting $\tilde{\Gamma}_{T-1} = \Gamma_{T-1}$ and for every $t \in H'$, we build $\tilde{\Gamma}_t = \{\overline{\boldsymbol{\alpha}}(\boldsymbol{b_t})|\boldsymbol{b_t} \in \tilde{B}_t\}$. We call $\tilde{V}_t$ the corresponding value function (according to Equation 3 where Γ_t, V_t and B are respectively replaced by $\tilde{\Gamma}_t$, $\tilde{V}_t$ and $\tilde{B}_t$). We have $\tilde{\Gamma}_t \subseteq \Gamma_t$, $t \in H$, thus every $\tilde{V}_t$ is potentially only a lower bound of V_t. If $\tilde{\Gamma}_{t+1} = \Gamma_{t+1}$ (i.e. $\tilde{V}_{t+1} = V_{t+1}$), then the maximal error on V_t is by definition $gap_t = \max_{\boldsymbol{b_t}\in B}(BL(\tilde{V}_{t+1})(\boldsymbol{b_t}) - \tilde{V}_t(\boldsymbol{b_t}))$. By induction from $\tilde{V}_{T-1} = V_{T-1}$, which does not have any error, and according to Lemma 1, the maximal error on any $\tilde{V}_t$ is $\sum_{t'\in\{t,\ldots,T-1\}} gap_{t'}$. Thus the maximal error on V_0 at $t = 0$ is bounded by $gap = \sum_{t\in H} gap_t$ for any $\boldsymbol{b_0}$. This measure of error is usual in approximation approaches [Hansen, 1998; Hauskrecht, 2000].

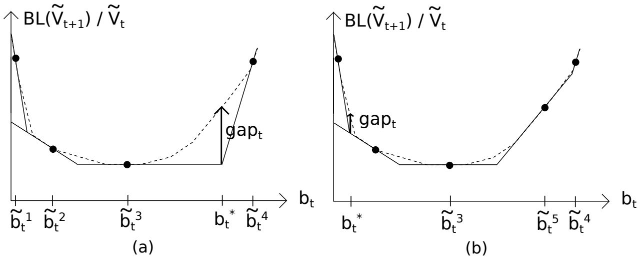
Lemma 1. *Let $t \in H'$ and $\tilde{V}_{t+1}$ be an approximate lower representation of V_{t+1} such that: $\forall\ \boldsymbol{b_{t+1}} \in B$, $\tilde{V}_{t+1}(\boldsymbol{b_{t+1}}) \leq$*

$V_{t+1}(\boldsymbol{b_{t+1}}) \leq \tilde{V}_{t+1}(\boldsymbol{b_{t+1}}) + \epsilon$. *Then,* $\forall\ \boldsymbol{b_t} \in B$, $\tilde{V}_t(\boldsymbol{b_t}) \leq V_t(\boldsymbol{b_t}) \leq \tilde{V}_t(\boldsymbol{b_t}) + \epsilon$, *where* $\tilde{V}_t = BL(\tilde{V}_{t+1})$ *and* $V_t = BL(V_{t+1})$.

Typical point-based methods sample belief states by simulating interactions with the environment and then updating the value function over a selection of those sampled belief states [Pineau *et al.*, 2006; Shani *et al.*, 2006]. Our approach consists, for every given time-step t, in searching iteratively for belief states $\boldsymbol{b_t^*}$ of Equation 6, in order to improve $\tilde{V}_t$.

$$\boldsymbol{b_t^*} = \underset{\boldsymbol{b_t} \in B}{\arg\max}[BL(\tilde{V}_{t+1})(\boldsymbol{b_t}) - \tilde{V}_t(\boldsymbol{b_t})] \tag{6}$$

The belief point set expansion consisting of adding $\boldsymbol{b_t^*}$ to $\tilde{B}_t$ and the improvement of $\tilde{V}_t$ consisting of adding $\overline{\boldsymbol{\alpha}}(\boldsymbol{b_t^*})$ to $\tilde{\Gamma}_t$ aim to reduce the current gap as much as possible iteratively (Figure 1).



(a)

(b)

Figure 1: Two successive iterations (a) and (b) of our method for a given time-step t. In both figures, solid lines represent $\tilde{V}_t$ while dashed lines represent $BL(\tilde{V}_{t+1})$, and belief $\boldsymbol{b_t^*}$ provides the current biggest gap between $BL(\tilde{V}_{t+1})$ and $\tilde{V}_t$. In (a), $\boldsymbol{\tilde{b}_t^1}$,..., $\boldsymbol{\tilde{b}_t^4}$ constitute the current set $\tilde{B}_t$. In these points, $BL(\tilde{V}_{t+1})$ and $\tilde{V}_t$ have the same values. In (b), the set $\tilde{B}_t$ is now equal to $\boldsymbol{\tilde{b}_t^1}$,..., $\boldsymbol{\tilde{b}_t^5}$. Adding $\boldsymbol{\tilde{b}_t^5}$ and its corresponding α-vector has reduced the gap between $BL(\tilde{V}_{t+1})$ and $\tilde{V}_t$.

Several algorithms have been proposed to approximately solve POMDPs. For example, Roy *et al.* [2005] aimed to reduce the dimensionality of the belief space using principal components analysis, rather than minimize the size of the solution itself. Poupart and Boutilier [2004] sought to reduce the dimensionality of the belief space and to apply the "bounded policy iteration" method [Poupart and Boutilier, 2003] based on bounded stochastic finite state controllers. In both cases [Poupart and Boutilier, 2004; Roy *et al.*, 2005], the methods do not provide an intrinsic performance guarantee, i.e. no gap is provided between the approximate and the optimal solutions. One must use an external algorithm to compute this gap.

Unlike point-based sampling approaches [Shani *et al.*, 2012] or other approximation approaches [Hauskrecht, 2000; Poupart and Boutilier, 2004; Roy *et al.*, 2005], our method generates iteratively new non-dominated α-vectors by solving Equation 6 directly.

4 A MILP approach

Given a time-step $t \in H'$, solving Equation 6 exactly is an NP-hard optimization problem. This is a direct consequence of the NP-hardness of the exact backup operation [Littman *et al.*, 1995]. Indeed, otherwise we could use Equation 6 to construct an exact representation of the optimal value function V_t given the exact value function V_{t+1}, in a time which is a polynomial in the instance and the number of α-vectors needed to describe V_t.

Equation 6 can be formulated as the following quadratic program QP:

$$\begin{array}{ll} \max & g_t \\ \text{s.t.} & g_t \leq \boldsymbol{\alpha_t} \cdot \boldsymbol{b_t} - \boldsymbol{\tilde{\alpha}_t} \cdot \boldsymbol{b_t}, \boldsymbol{\tilde{\alpha}_t} \in \tilde{\Gamma}_t \\ & \boldsymbol{b_t} \cdot \boldsymbol{1} = 1 \\ & \boldsymbol{b_t} \geq \boldsymbol{0} \\ & \boldsymbol{\alpha_t} \in P(\tilde{\Gamma}_{t+1}) \end{array} \tag{QP}$$

where the variables are $\boldsymbol{\alpha_t}$ and $\boldsymbol{b_t}$. The first term $\boldsymbol{\alpha_t} \cdot \boldsymbol{b_t}$ of the first constraints corresponds to the α-vector formulation of $BL(\tilde{V}_{t+1})(\boldsymbol{b_t})$ and $\boldsymbol{\tilde{\alpha}_t} \cdot \boldsymbol{b_t}$ corresponds to the α-vector formulation of $\tilde{V}_t(\boldsymbol{b_t})$. Note that $\boldsymbol{\tilde{\alpha}_t}$ are not variables but coefficients of the $|\tilde{\Gamma}_t|$ first constraints.the first type.

The problem QP is difficult to solve, not only because it is a non-concave quadratic program, but mainly because $P(\tilde{\Gamma}_{t+1})$ is not known. In the remainder of the paper, we will demonstrate how we can reformulate QP into a MILP that can be solved efficiently.

Because the last constraint $\boldsymbol{\alpha_t} \in P(\tilde{\Gamma}_{t+1})$ is not expressed as a set of linear inequalities, we have still to find a set of such linear inequalities describing $Conv_t$, the convex hull of $P(\tilde{\Gamma}_{t+1})$. Since we are in a maximization case, we can focus on describing C_t, the polyhedron composed of hyperplanes of $Conv_t$ with positive normal vectors and hyperplanes $\alpha_{t,s} = 0, s \in S$, where $\alpha_{t,s}$ are the components of $\boldsymbol{\alpha_t}$. Note that extreme points of C_t with strictly positive coordinates correspond to non-dominated α-vectors of $P(\tilde{\Gamma}_{t+1})$, while interior points of $P(\tilde{\Gamma}_{t+1})$ correspond to dominated α-vectors.

Let us describe C_t using an infinite number of constraints: $C_t = \{\boldsymbol{\alpha_t} \in \mathbb{R}^{|S|} \mid \boldsymbol{\alpha_t} \cdot \boldsymbol{b_t} \leq \overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\boldsymbol{b_t}) \cdot \boldsymbol{b_t}, \boldsymbol{b_t} \in B, \boldsymbol{\alpha_t} \geq \boldsymbol{0}\}$. We can approximate C_t with a finite number of constraints by considering the convex polyhedron $\tilde{C}_t^n = \{\boldsymbol{\alpha_t} \in \mathbb{R}^{|S|} \mid \boldsymbol{\alpha_t} \cdot \boldsymbol{\mathring{b}_t^i} \leq \overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\boldsymbol{\mathring{b}_t^i}) \cdot \boldsymbol{\mathring{b}_t^i}, i = 0, ..., n-1, \boldsymbol{\alpha_t} \geq \boldsymbol{0}\}$ where $\boldsymbol{\mathring{b}_t^0}, ..., \boldsymbol{\mathring{b}_t^{n-1}}$ are $n \geq |S|$ beliefs of B. We have $C_t \subseteq \tilde{C}_t^n$.

For a given C_t^n, instead of solving QP, we can now solve the easier quadratic program $QP_n(\boldsymbol{\mathring{b}_t^1}, ..., \boldsymbol{\mathring{b}_t^{n-1}})$, which can provide an approximation of QP with guaranteed performance (Proposition 2):

$$\begin{array}{ll} \max & g_t^n \\ \text{s.t.} & g_t^n \leq \boldsymbol{\alpha_t^n} \cdot \boldsymbol{b_t^n} - \boldsymbol{\tilde{\alpha}_t} \cdot \boldsymbol{b_t^n}, \boldsymbol{\tilde{\alpha}_t} \in \tilde{\Gamma}_t \\ & \boldsymbol{b_t^n} \cdot \boldsymbol{1} = 1 \\ & \boldsymbol{b_t^n} \geq \boldsymbol{0} \\ & \boldsymbol{\alpha_t^n} \in \tilde{C}_t^n \end{array} \tag{QP_n}$$

Proposition 2. *Let $\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\hat{\tilde{\boldsymbol{\alpha}}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, and $\hat{g}_t^n$ be an optimal solution of QP_n.*

At the optimum of QP_n, we have $\max_{\boldsymbol{b_t} \in B}(BL(\tilde{V}_{t+1})(\boldsymbol{b_t}) - \tilde{V}_t(\boldsymbol{b_t})) \leq \hat{g}_t^n \leq \max_{\boldsymbol{b_t} \in B}(BL(\tilde{V}_{t+1})(\boldsymbol{b_t}) - \tilde{V}_t(\boldsymbol{b_t})) + \delta_t^n$, where $\delta_t^n = \hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}} \cdot \hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}} - \overline{\boldsymbol{\alpha}}(\hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}) \cdot \hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}$.

This formulation allows us to decide whether to expand the belief set by adding $\hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}$ to $\tilde{B}_t$ in the case where δ_t^n is small enough, or to construct, from the current solution of QP_n, a new relevant belief $\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}$ defining a new constraint of $\tilde{C}_t^n$, in order to get a better approximation $\tilde{C}_t^{n+1}$ of C_t.

Our aim is then to find a good description of C_t, i.e. using few hyperplanes. It is well known in linear programming, and particularly in polyhedral approaches, that the best possible hyperplanes describing a convex polyhedron correspond to its so-called *facets* [Mahjoub, 2014]: the set of these facets is indeed the minimal set of hyperplanes needed to describe a convex polyhedron and is always finite. The problem of finding relevant facets is known as *separation problem* [Mahjoub, 2014]. Grötschel *et al.* [1981] showed that the cost of optimization on a given polyhedron does not depend on the number of constraints of the system describing the polyhedron, but rather on the separation problem associated with this system. In our case, optimizing over C_t means finding a new relevant non-dominated α-vector and its associated belief. Unfortunately, since QP_n is quadratic, one cannot directly apply the results of Grötschel *et al.* [1981] to solve QP_n.

An important step in proving the convergence of our proposed algorithms is to solve the following *separation problem*, formally: compute a new facet F_t^n of C_t from any current solution $\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$ of QP_n which is not already in C_t. This can be done by using the algorithm GenerateFacet($\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\tilde{\Gamma}_{t+1}$) (Proposition 3).

Proposition 3. *Given a set $\tilde{\Gamma}_{t+1}$ of α-vectors at time-step t representing the value function $\tilde{V}_{t+1}$, and a vector $\boldsymbol{\alpha_t} \in \mathbb{R}^{|S|}$, one can decide if $\boldsymbol{\alpha_t}$ belongs to C_t or not, and if not, generate a facet of C_t. We call the corresponding algorithm $GenerateFacet(\boldsymbol{\alpha_t}, \tilde{\Gamma}_{t+1})$.*

Each time we solve QP_n and δ_t^n is not satisfying, we can generate a new facet F_t^n from $\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$ using $GenerateFacet(\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}, \tilde{\Gamma}_{t+1})$. We then can set $\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}} = \boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{F_t^n}}$ to add the corresponding constraint to $\tilde{C}_t^n$ (which becomes $\tilde{C}_t^{n+1}$), where $\boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{F_t^n}}$ is the belief corresponding to the facet F_t^n. In doing this, $\tilde{C}_t^n$ converges to C_t (Proposition 4).

Proposition 4. *$\tilde{C}_t^n$ converges to C_t since at each iteration we compute a new facet of C_t, and the convex polyhedron C_t has a finite number of facets. Thus, there exists $n^* \in \mathbb{N}$ such that the solution $\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n^*}}$ of QP_{n^*} belongs to C_t, i.e. $\delta_t^{n^*} = 0$. Therefore, for a given precision $\epsilon_p > 0$ there exists $n_{\epsilon_p} \in \mathbb{N}$ such that $\delta_t^{n_{\epsilon_p}} \leq \epsilon_p$.*

QP_n is however still not easy to solve, because the objective function is clearly not concave, due to the term $\boldsymbol{\alpha_t} \cdot \boldsymbol{b_t}$. But, since $\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$ necessarily belongs to one of the extreme points of $\tilde{C}_t^n$, it satisfies the System 7:

$$\begin{aligned} &\boldsymbol{\alpha_t} \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}} \leq \overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}) \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}},\ i = 0, ..., n-1 \\ &\boldsymbol{\alpha_t} \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}} = \overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}}) \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}},\ j = 1, ..., |S| \end{aligned} \tag{7}$$

Let $\beta_1, ..., \beta_{|S|} \in [0, 1]$ such that $\hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}} = \sum_{j \in 1,...,|S|} \beta_j \mathring{\boldsymbol{b}}_{\boldsymbol{t}}{}^{i_j}$ and $\sum_{j \in 1,...,|S|} \beta_j = 1$. Such β_j and $\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}}$ always exist if we assume that there are at least $|S|$ beliefs $\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}$ affinely independent (for example, by setting the $|S|$ first $\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}$ to the extreme points of the simplex B). $\boldsymbol{\alpha_t} \cdot \boldsymbol{b_t}$ can be easily re-written $\sum_{j=1,...,|S|} \beta_{i_j} (\overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}}) \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i_j}})$, which is a linear expression. This technique of linearization is similar to considering interpolations of $BL(\tilde{V}_{t+1})$ as an upper bound [Poupart *et al.*, 2011]. Our approach differs from [Poupart *et al.*, 2011] because we calculate a tight upper bound in order to guarantee a near best possible improvement of the lower bound instead of computing upper bound and lower bound with two independent heuristics.

Thus, we can finally reformulate QP_n into a MILP (Proposition 5).

Proposition 5. *Let $\boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{0}}, ..., \boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{n-1}}$ be $n \geq |S|$ beliefs. Problem $QP_n(\boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{0}}, ..., \boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{n-1}})$ can be reformulated as the following MILP, called $MILP_n(\boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{0}}, ..., \boldsymbol{b}_{\boldsymbol{t}}^{\boldsymbol{n-1}})$, containing only continuous variables except n that are 0-1 variables.*

$$\begin{aligned} \textit{max} \quad & g_t \\ \textit{s.t.} \quad & g_t \leq W_t - U_t \\ & W_t = \textstyle\sum_{i=1,...,n} \beta_i (\overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}) \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}) \\ & \textstyle\sum_{j=1,...,|S|} \beta_j = 1 \\ & \boldsymbol{b_t} \cdot \boldsymbol{1} = 1 \\ & \boldsymbol{b_t} = \textstyle\sum_{i=1,...,n} \beta_i \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}} \\ & \boldsymbol{\alpha_t} \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}} + y_i = \overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}(\mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}) \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}, i = 0, ..., n-1 \\ & y_i \leq M(1 - x_i), i = 0, ..., n-1 \\ & U_t \geq \tilde{\boldsymbol{\alpha}}_{\boldsymbol{t}} \cdot \boldsymbol{b_t}, \tilde{\boldsymbol{\alpha}}_t \in \tilde{\Gamma}_t \\ & \beta_i \leq x_i, i = 0, ..., n-1 \\ & \textstyle\sum_{i=0,...,n-1} x_i \leq |S| \\ & \boldsymbol{b_t} \geq \boldsymbol{0} \\ & y_i \geq 0, i = 0, .., n-1 \\ & \beta_i \leq 1, i = 0, .., n-1 \\ & \beta_i \geq 0, i = 0, .., n-1 \\ & x_i \in \{0, 1\}, i = 0, .., n-1 \\ & W_t, U_t, g_t \geq 0 \end{aligned} \tag{$MILP_n$}$$

where $M \geq T \times R_{max}$ with R_{max} the maximum reward over the matrix R. M is then an upper bound of the term $\boldsymbol{\alpha_t} \cdot \mathring{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{i}}$.

Any optimal solution of $MILP_n$ is also an optimal solution of QP_n. We keep the same notation for the solutions: ($\hat{\boldsymbol{\alpha}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\hat{\tilde{\boldsymbol{\alpha}}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\hat{\boldsymbol{b}}_{\boldsymbol{t}}^{\boldsymbol{n}}$, $\hat{g}_t^n$).

We now have all the tools to solve Equation 6 with bounded error (Proposition 6). The procedure to find the corresponding near optimal belief (FBB) is given in Algorithm 1.

Proposition 6. *Given a time-step t, $\tilde{V}_t$, $\tilde{V}_{t+1}$ and a representation $\tilde{B}_t$ of the belief space B, one can solve approximately the optimization problem corresponding to Equation 6 to within specified precision ϵ_p, using Algorithm 1, called FBB (Find Best Belief).*

Algorithm 1 Find the best belief for expanding $\tilde{B}_t$ according to Equation 6 to within specified precision ϵ_p

```
1:  procedure FBB($t$, $\tilde{B}_t$, $\tilde{\Gamma}_t$, $\tilde{\Gamma}_{t+1}$, $\epsilon_p$)
2:      $n \leftarrow 0$
3:      for $s \in S$ do
4:          $\mathring{\boldsymbol{b}}_t^n \leftarrow \boldsymbol{e_s}$          ▷ $\boldsymbol{e_s}, s \in S$ are extreme points of $B$
5:          $n \leftarrow n + 1$
6:      while $\Delta > \epsilon_p$ do
7:          $(\hat{\boldsymbol{\alpha}}_t^n, \hat{\boldsymbol{b}}_t^n, \hat{g}_t^n) \leftarrow$ Solve $MILP_n(\mathring{\boldsymbol{b}}_t^0, ..., \mathring{\boldsymbol{b}}_t^{n-1})$
8:          $\Delta \leftarrow \delta_t^n = \hat{\boldsymbol{\alpha}}_t^n \cdot \hat{\boldsymbol{b}}_t^n - \overline{\boldsymbol{\alpha}}(\hat{\boldsymbol{b}}_t^n) \cdot \hat{\boldsymbol{b}}_t^n$
9:          $F_t^n \leftarrow$ GenerateFacet($\hat{\boldsymbol{\alpha}}_t^n, \tilde{\Gamma}_{t+1}$)
10:         $\mathring{\boldsymbol{b}}_t^n \leftarrow \boldsymbol{b}_t^{F_t^n}$
11:         $n \leftarrow n + 1$
        return $\hat{g}_t^n, \hat{\boldsymbol{b}}_t$
```

Proposition 7. *In the worst case, Algorithm 1 requires $O(P_N \times 2^{N+|S|+1})$ operations, where N designates the maximum number of facets needed to describe C_t and P_N is a polynomial in N and the size of the POMDP.*

Theorem 8. *In finite time, one can solve approximately any finite-horizon POMDP with a specified arbitrary maximum gap ϵ, using Algorithm 2 by setting $\epsilon_p \leq \epsilon$.*

Algorithm 2, which we call ϵ-min, aims to provide a compact solution under the constraint of respecting a gap less than or equal to a fixed parameter ϵ uniformly spread across each time-step. ϵ-min is attractive because one can specify a required maximum gap, but ϵ-min may lead naturally to the calculation of a large number of α-vectors per time-step, since the algorithm adds new α-vectors until the required maximum gap is reached. This behavior is not suitable when looking for simple solutions.

However, we can easily adapt ϵ-min to constrain the maximum number of α-vectors to use per time-step. We call this algorithm α-min (Algorithm 3).

Theorem 9. *In finite time, one can solve approximately any finite-horizon POMDP with a specified arbitrary maximum number N of α-vectors per time-step using Algorithm 3 by setting ϵ_p small enough. Algorithm 3 provides a maximum gap between the solution it computes and an optimal solution.*

Algorithm 2 ϵ-min: Solve POMDP with a maximum gap ϵ, to within precision ϵ_p

```
1:  procedure ε-MIN($H$, $A$, $S$, $O$, $\epsilon$, $\epsilon_p$)
2:      $\tilde{\Gamma}_{T-1} \leftarrow \{\boldsymbol{r_a} |\, a \in A,\ \boldsymbol{r_a}$ is not dominated$\}$
3:      $gap_{T-1} = 0$
4:      for $t \in H'$ do
5:          $\tilde{B}_t = \boldsymbol{b}_t^{init}$          ▷ $\boldsymbol{b}_t^{init}$ is an arbitrary belief
6:          $gap_t \leftarrow \infty$
7:      for $t = T-2, T-3, ..., 0$ do
8:          while $gap_t > \frac{\epsilon}{T-1}$ do
9:              $(gap_t, \boldsymbol{b}_t^*) \leftarrow$ FBB($t, \tilde{B}_t, \tilde{\Gamma}_t, \tilde{\Gamma}_{t+1}, \frac{\epsilon_p}{T-1}$)
10:             $\tilde{B}_t \leftarrow \tilde{B}_t \cup \{\boldsymbol{b}_t^*\}$
11:             $\tilde{\Gamma}_t \leftarrow \tilde{\Gamma}_t \cup \{\overline{\boldsymbol{\alpha}}(\boldsymbol{b}_t^*)\}$
12:     $gap \leftarrow \sum_{t \in H'} gap_t$
13:     return $\{\tilde{\Gamma}_t, t \in H\}, gap$
```

Algorithm 3 α-min: Solve POMDP with a maximum number N of α-vectors, to within precision ϵ_p

```
1:  procedure α-MIN($H$, $A$, $S$, $O$, $N$, $\epsilon_p$)
2:      Lines 2 to 7 of Algorithm 2
3:      while $|\tilde{\Gamma}_t| < N$ and $gap_t > \frac{\epsilon_p}{T-1}$ do
4:      Lines 9 to 13 of Algorithm 2
```

5 Experiments

We first assess the performance of α-min on four small finite-horizon POMDP problems from the literature[2]and a larger randomly generated problem (random30). We compare its performance to a leading infinite-horizon POMDP solver Sarsop [Kurniawati *et al.*, 2008]. Sarsop results were obtained by solving the POMDPs over an infinite-horizon with $\gamma = 0.999$ and a maximum computational time of 1000s. Sarsop lower bounds (LB) were calculated as the expected sum of rewards cumulated over T time-steps by simulation of the infinite policy using a γ=1. α-min results were obtained using a fixed number of α-vectors set arbitrarily with a maximum computational time of 1000s per time-step on a 94.4 GB, 3.47GHz, 19 cores computer and CPLEX 12.5. Overall, the performance of α-min is encouraging with surprisingly good gaps obtained considering the small quantity of α-vectors (Table 1). The lower bounds of Sarsop and α-min are close. Table 2 shows the behavior of the lower bounds and gaps generated by α-min as the allowed number of α-vectors is increased for the problem milos-aaai97.

Table 1: For each T, this table reports for Sarsop and α-min the number of α-vectors, the lower bound and gap achieved.

Problem	**Algo.**	$\boldsymbol{|\alpha|}$	**LB** T=10	**LB** T=20	**LB** T=30	**LB** T=∞
			(gap)	(gap)	(gap)	(gap)
aloha.10	sarsop	190	**64.87**	**89.09**	**95.77**	535.46
$\lvert S\rvert = 30$						(9.03)
$\lvert A\rvert = 9$	α-min	**30**	62.66	85.46	94.53	
$\lvert O\rvert = 3$			(6.67)	(20.97)	(38.22)	
	$\frac{\alpha\text{-min}}{\text{sarsop}}$		0.966	0.96	0.987	
learning.c3	sarsop	11433	1.36	**2.29**	**2.36**	2.45
$\lvert S\rvert = 24$						(0.207)
$\lvert A\rvert = 12$	α-min	**24**	**1.96**	2.11	2.16	
$\lvert O\rvert = 3$			(6.23)	(13.26)	(18.34)	
	$\frac{\alpha\text{-min}}{\text{sarsop}}$		1.441	0.921	0.915	
cheng.D4-5	sarsop	15	77.29	153.03	232.55	7883.85
$\lvert S\rvert = 4$						(92.85)
$\lvert A\rvert = 4$	α-min	**4**	**77.85**	**156.55**	**235.25**	
$\lvert O\rvert = 4$			(1.61)	(3.59)	(5.57)	
	$\frac{\alpha\text{-min}}{\text{sarsop}}$		1.007	1.023	1.011	
milos-aaai97	sarsop	122	41.48	**99.75**	**176.07**	5801.19
$\lvert S\rvert = 20$						(6399.01)
$\lvert A\rvert = 6$	α-min	**20**	**50.31**	89.20	140.23	
$\lvert O\rvert = 8$			(104.62)	(419.66)	(608.39)	
	$\frac{\alpha\text{-min}}{\text{sarsop}}$		1.213	0.894	0.796	
random30	sarsop	29	598.21	**1215.99**	**1811.95**	603.43
$\lvert S\rvert = 60$						(38.06)
$\lvert A\rvert = 30$	α-min	**10**	**599.04**	1198.74	1798.16	
$\lvert O\rvert = 30$			(418.51)	(942.95)	(1530.21)	
	$\frac{\alpha\text{-min}}{\text{sarsop}}$		1.001	0.985	0.992	

[2] http://pomdp.org/examples/index.shtml

Table 2: Improvement of the lower bound and the gap as the number of α-vectors is increased.

problem	**algorithm**	$\lvert\boldsymbol{\alpha}\rvert$	**LB** T=10	**gap** T=10
milos-aaai97	α-min	6	44.98	124.78
milos-aaai97	α-min	10	46.67	115.50
milos-aaai97	α-min	15	49.75	104.68
milos-aaai97	α-min	20	50.31	104.62
milos-aaai97	α-min	30	50.73	80.40

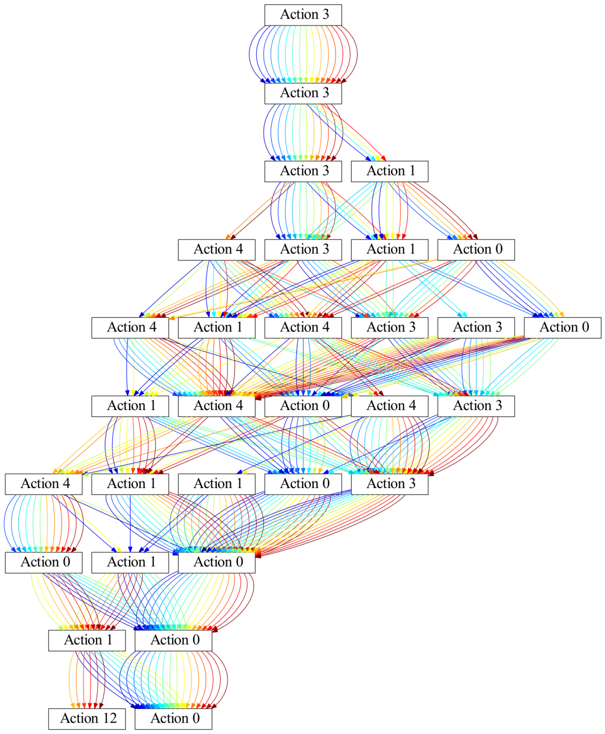


Figure 2: Policy graph of the four populations Sumatran tigers non stationary problem assuming the starting belief state of all populations 'extant'. Each color corresponds to one possible observation.

Finally, we illustrate the benefits of using α-min to best manage or monitor four sub-populations of Sumatran tigers with declining connectivity over a 10-year time horizon [Linkie *et al.*, 2006]. We model this problem as a non-stationary finite-horizon POMDP with 16 states representing the status extinct or extant of each sub-population; 13 actions representing the decisions of doing nothing, managing and/or monitoring each sub-population; 16 observations representing the difficulty of detecting Sumatran tigers in each sub-population (absent or present); and 10 transition matrices representing the declining connectivity between subpopulations over time. The probabilities of going extinct with and without management were derived based on tiger census estimates [Chadès *et al.*, 2008; Linkie *et al.*, 2006]. Detection probabilities were derived based on [McDonald-Madden *et al.*, 2011], and projected fragmentation scenarios followed [Linkie *et al.*, 2006]. Interested readers can refer to the supplementary material[3] for the POMDP files corresponding to this problem and to [Chadès *et al.*, 2008; McDonald-Madden *et al.*, 2011; Regan *et al.*, 2011] for limitations and advantages of using POMDPs in conservation problems. For sake of illustration, Figure 2 presents the policy graph obtained for T=10 and $|\alpha|$=7 by α-min. In this case, the solution is guaranteed to be at most 10% from the optimal strategy ($gap < \frac{LB}{10}$). The CPU time required to solve the problem was 378 seconds. Note that because this is a non-stationary finite-horizon POMDP, it is not possible to provide a Sarsop solution.

6 Discussion

We proposed two new algorithms, ϵ-min and α-min, to approximately solve finite-horizon POMDPs. Both algorithms rely on finding at each time-step α-vectors to minimize the gap between the optimal value function and its current approximation, as formally expressed by Equation 6. ϵ-min calculates an approximate solution given a maximum gap, which can lead to the calculation of very many α-vectors. α-min is more directly applicable to computational sustainability as it provides good POMDP solutions given a fixed number of α-vectors. α-min can easily be adapted to find approximate solutions given a CPU time limit, or both a time limit and a maximum number of α-vectors.

α-min greedily expands the set of beliefs and the set of α-vectors by adding iteratively new beliefs and α-vector which aim to reduce the current gap as much as possible, until N α-vectors have been added. One interesting possible future direction would be to improve α-min in order to generate a set of α-vectors of a given cardinality N with the guarantee that no strictly-better set of the same cardinality exists (to within a given precision).

Unlike most point-based approaches, our algorithms assume that we do not know the initial belief $\boldsymbol{b_0}$. Taking advantage of $\boldsymbol{b_0}$ could be explored in future work.

Our method can also be adapted to the infinite-horizon case, since it is a point-based approach. In our case, it was particularly interesting to propose a solver for finite-horizon POMDPs as it allows us to generate non-stationary policies, given that, in the context of computational sustainability, the transition matrices and rewards might change over time.

Finally, the complexity bound of Proposition 7 could probably be improved. However finding an "efficient" complexity bound, e.g. polynomial in the instance, is unlikely given the non-approximability results for POMDPs in general and for finite-horizon POMDPs in particular [Lusena *et al.*, 2001].

Notation	**Description**
$\tilde{V}_t$	Lower bound of the value function V_t
$\tilde{\Gamma}_t$	Set of α-vectors describing $\tilde{V}_t$
$\tilde{\boldsymbol{\alpha}}_{\boldsymbol{t}}$	α-vector belonging to $\tilde{\Gamma}_t$
BL	Bellman operator
$\overline{\boldsymbol{\alpha}}_{\boldsymbol{t}}$	Such that $\forall \boldsymbol{b_t} \in B, \overline{\boldsymbol{\alpha}}(\boldsymbol{b_t}) \cdot \boldsymbol{b_t} = BL(\tilde{V}_{t+1})(\boldsymbol{b_t})$

Table 3: Main notations table.

[3] https://sites.google.com/site/ijcaialphamin/home

References

[Bellman, 1957] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[Chadès *et al.*, 2008] I. Chadès, E. McDonald-Madden, M. A. McCarthy, B. Wintle, M. Linkie, and H. P. Possingham. When to stop managing or surveying cryptic threatened species. *Proceedings of the National Academy of Sciences*, 105(37):13936–13940, 2008.

[Chadès *et al.*, 2011] I. Chadès, T. G. Martin, S. Nicol, M. A. Burgman, H. P. Possingham, and Y. M. Buckley. General rules for managing and surveying networks of pests, diseases, and endangered species. *Proceedings of the National Academy of Sciences*, 108(20):8323–8328, 2011.

[Grötschel *et al.*, 1981] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[Hansen, 1998] E. Hansen. *Finite-memory control of partially observable MDPs*. PhD thesis, University of Massachussets at Amherst, 1998.

[Hauskrecht, 2000] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

[Kurniawati *et al.*, 2008] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, volume 2008. Zurich, Switzerland, 2008.

[Linkie *et al.*, 2006] M. Linkie, G. Chapron, D. J. Martyr, J. Holden, and N. Leader-Williams. Assessing the viability of tiger subpopulations in a fragmented landscape. *Journal of Applied Ecology*, 43(3):576–586, 2006.

[Littman *et al.*, 1995] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Efficient dynamic-programming updates in partially observable Markov decision processes. Technical report, 1995.

[Lusena *et al.*, 2001] C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable Markov decision processes. *J. Artif. Intell. Res.(JAIR)*, 14:83–103, 2001.

[Madani *et al.*, 2003] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1):5–34, 2003.

[Mahjoub, 2014] A. R. Mahjoub. *Concepts of Combinatorial Optimization*, chapter Polyhedral Approaches. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2014.

[McDonald-Madden *et al.*, 2011] E. McDonald-Madden, I. Chadès, M. A McCarthy, M. Linkie, and H. P. Possingham. Allocating conservation resources between areas where persistence of a species is uncertain. *Ecological Applications*, 21(3):844–858, 2011.

[Nicol and Chadès, 2012] S. Nicol and I. Chadès. Which states matter? An application of an intelligent discretization method to solve a continuous POMDP in conservation biology. *PLoS ONE*, 7(2):e28993, 02 2012.

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

[Pineau *et al.*, 2006] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, pages 335–380, 2006.

[Poupart and Boutilier, 2003] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in neural information processing systems*, page None, 2003.

[Poupart and Boutilier, 2004] P. Poupart and C. Boutilier. Vdcbpi: an approximate scalable algorithm for large pomdps. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2004.

[Poupart *et al.*, 2011] P. Poupart, K.-E. Kim, and D. Kim. Closing the gap: Improved bounds on optimal POMDP solutions. In *ICAPS*, 2011.

[Åström, 1965] K. J. Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174, 1965.

[Regan *et al.*, 2011] T. J Regan, I. Chadès, and H. P. Possingham. Optimally managing under imperfect detection: a method for plant invasions. *Journal of Applied Ecology*, 48(1):76–85, 2011.

[Roy *et al.*, 2005] N. Roy, G. J. Gordon, and S. Thrun. Finding approximate pomdp solutions through belief compression. *J. Artif. Intell. Res.(JAIR)*, 23:1–40, 2005.

[Shani *et al.*, 2006] G. Shani, R. I Brafman, and S. E. Shimony. Prioritizing point-based POMDP solvers. In *Machine Learning: ECML 2006*, pages 389–400. Springer, 2006.

[Shani *et al.*, 2012] G. Shani, J. Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, June 2012.

[Sigaud and Buffet, 2013] O. Sigaud and O. Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[Smallwood and Sondik, 1973] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon . *Operations Research*, 21(5):1071–1088, 1973.

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