\(\alpha\text{-min: A Compact Approximate Solver For Finite-Horizon POMDPs}\)

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 \(\alpha\)-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 \(\alpha\)-vectors. This paper studies methods that allow the user to explore the tradeoff between the accuracy of the value function and the number of \(\alpha\)-vectors. Building on previous point-based POMDP solvers, this paper introduces a new algorithm (\(\alpha\)-min) that formulates a Mixed Integer Linear Program (MILP) to calculate approximate solutions for finite-horizon POMDP problems with limited numbers of \(\alpha\)-vectors. At each time-step, \(\alpha\)-min calculates \(\alpha\)-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 \(\alpha\)-vectors. Experimental results show that \(\alpha\)-min provides good approximate solutions given a fixed number of \(\alpha\)-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 \(\alpha\)-min, a finite-horizon POMDP solver that calculates a good policy given a limit on the number of \(\alpha\)-vectors.

Section 2 provides an overview of POMDPs. Section 3 introduces the principles of our approach that relies on calculating \(\alpha\)-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 \(\epsilon\text{-min}\) that finds approximate solutions given a fixed maximum gap. From \(\epsilon\text{-min}\), we then derive \(\alpha\)-min that finds good approximate solutions given a fixed number of \(\alpha\)-vectors. Section 5 assesses the performance of \(\alpha\)-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 \(\{S, A, O, \tau, \Omega, \tau, R, H\}\), where

- \(H = \{0,\ldots,T - 1\}, T \in \mathbb{N}\), 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
• \( \tau_a \) is the transition matrix for action \( a \). Elements are \( \tau_a(s_t, s_{t+1}) \).

• \( \Omega_a \) is the observation matrix for action \( a \). Elements are \( \Omega_a(s_{t+1}, z_{t+1}) \).

• \( R \) is the reward matrix. Elements are \( R(a_t, s_t) \).

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 \( r_a = R(a, \cdot) \) 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 [Aström, 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 \( \pi_t : B \rightarrow A \) maps belief states at time \( t \) to actions. \( \Pi_0 \) maximizes the expected sum of rewards \( E[\sum_{t \in H} r_{a_t} \cdot b_t] \) over the time horizon \( H \) (denotes the scalar product). For each time-step \( t \), for a given belief state \( b_t \) and a given policy \( \Pi_t = \cup_{t \in \{t, ..., T-1\}} \pi_t \), the expected sum \( E[\sum_{t \in \{t, ..., T-1\}} r_{a_t} \cdot b_t] \) is also referred to as the value function \( V_{t+1}(b_t) \). A value function allows us to rank strategies by assigning a real value to each belief \( b_t \). An optimal policy \( \Pi_t \) is a policy such that, \( \forall b_t \in B \), \( \forall \Pi_{t+1}, V_{t+1}(b_t) \geq \Pi_{t+1}(b_t) \). 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]: \( \forall b_t \in B \),

\[
V_t(b_t) = \max_{a_t \in A} \left\{ r_{a_t} \cdot b_t + \sum_{z_t \in O} p(z_{t+1}|a_t, b_t) V_{t+1}(b_{t+1}) \right\}
\] (1)

where the belief \( b_{t+1} \) can be computed as follows:

\[
b_{t+1}(s_{t+1}) = \frac{\Omega_a(s_{t+1}, z_{t+1}) \sum_{s_t, s'_{t+1}} \tau_a(s_t, s_{t+1}) b_t(s_t) \sum_{s_t, s'_{t+1}} \Omega_a(s'_{t+1}, z_{t+1}) \tau_a(s_t, s'_{t+1}) b_t(s_t)}{\sum_{s_t, s'_{t+1}} \sum_{s_t, s'_{t+1}} \Omega_a(s'_{t+1}, z_{t+1}) \tau_a(s_t, s'_{t+1}) b_t(s_t)}
\] (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 the 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 PSPACOMPLETE [Papadimitriou and Tsitsiklis, 1987] and infinite-horizon POMDPs are undecidable [Madani et al., 2003].

3 \( \alpha \)-min principle

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

\[
\forall t \in H, \forall b_t \in B, V_t(b_t) = \max_{\alpha_t \in \Gamma_t} \alpha_t \cdot b_t
\] (3)

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

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

where \( a_t \in A \) and \( \alpha_{t+1}^{a_t, z_{t+1}} \), \( z_{t+1} \in O \) are elements of \( \Gamma_{t+1} \).

The set of \( \alpha \)-vectors \( P(\Gamma_{t+1}) \) given by Equation 4 is such that \( \Gamma_t \subseteq P(\Gamma_{t+1}) \) (\( \alpha \)-vectors of \( P(\Gamma_{t+1}) \) potentially belong to \( \Gamma_t \)). In the case of exact resolution, for a given \( \Gamma_{t+1} \), a natural way to compute \( \Gamma_t \) 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 \( B_t \subseteq B \) sampled to be as relevant as possible to get good approximations of \( \Gamma_t \) at each time-step. Every alpha vector \( \alpha_{t}^{b_t} \) of the current approximation \( \Gamma_t \subseteq \Gamma_t \), corresponding to the belief \( b_t \in B_t \), is generated as follows:

\[
\alpha_{t}^{b_t} = \arg\max_{\alpha_t \in P(\Gamma_{t+1})} \alpha_t \cdot b_t
\] (5)

where \( \Gamma_{t+1} \) is an approximation of \( \Gamma_{t+1} \).

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

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 \( \tilde{\Gamma}_{t} = \{\overline{\alpha}(b_t) | b_t \in B_t \} \). We call \( \tilde{V}_t \) the corresponding value function (according to Equation 3 where \( \Gamma_t, V_t \) and \( B_t \) 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., \( V_{t+1} = \tilde{V}_{t+1} \)), then the maximal error on \( V_t \) is by definition \( gap_t = \max_{b_t \in B} (BL(V_{t+1})(b_t) - \tilde{V}_t b_t) \). By induction from \( \tilde{V}_{T-1} = \tilde{V}_{T-1} \), which does not have any error, and according to Lemma 1, the maximal error on any \( \tilde{V}_t \) is \( \sum_{t \in \{t, ..., T-1\}} gap_t \). Thus the maximal error on \( \tilde{V}_0 \) at \( t = 0 \) is bounded by \( gap = \sum_{b_0} gap_b \) for any \( b_0 \). 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: \( \forall b_{t+1} \in B, \tilde{V}_{t+1}(b_{t+1}) \leq \tilde{V}_t b_t \leq \tilde{V}_{t+1}(b_{t+1}) \).
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 $\tilde{b}_t^*$ of Equation 6, in order to improve $\tilde{V}_t$.

$$b^*_t = \arg \max_{b_t \in B} [BL(\tilde{V}_{t+1})(b_t) - \tilde{V}_t(b_t)]$$

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

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

where the variables are $\alpha_t$ and $b_t$. The first term $\alpha_t \cdot b_t$ of the first constraints corresponds to the $\alpha$-vector formulation of $BL(\tilde{V}_{t+1})(b_t)$ and $\tilde{\alpha}_t \cdot b_t$ corresponds to the $\alpha$-vector formulation of $\tilde{V}_t(b_t)$. Note that $\tilde{\alpha}_t$ 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 $\alpha_t \in P(\tilde{\Gamma}_{t+1})$ 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 $\alpha_t$. Note that extreme points of $C_t$ with strictly positive coordinates correspond to non-dominated $\alpha$-vectors of $P(\tilde{\Gamma}_{t+1})$, while interior points of $P(\tilde{\Gamma}_{t+1})$ correspond to dominated $\alpha$-vectors.

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

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

$$\begin{align*}
\text{max} & \quad g^n_t \\
\text{s.t.} & \quad g^n_t \leq \alpha_t^n \cdot b_t^n - \tilde{\alpha}_t^n \cdot b_t^n, \tilde{\alpha}_t^n \in \tilde{\Gamma}_t \\
b^n_t \cdot 1 = 1 \\
b^n_t \geq 0 \\
\alpha_t^n \in C^n_t
\end{align*}$$
Proposition 2. Let \( \hat{\alpha}_t^n, \hat{\beta}_t^n, \tilde{\alpha}_t^n \) and \( \tilde{\gamma}_t^n \) be an optimal solution of \( QP_n \).

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

This formulation allows us to decide whether to expand the belief set by adding \( \hat{b}_t^n \) to \( B_t \), in the case where \( \delta_t^n \) is small enough, or to construct, from the current solution of \( QP_n \), a new relevant belief \( \hat{b}_t^n \) defining a new constraint of \( \tilde{C}_t^n \), in order to get a better approximation \( \tilde{C}_{t+1} \) 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 \( \alpha \)-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 \( \hat{\alpha}_t^n \) of \( QP_n \), which is not already in \( C_t \). This can be done by using the algorithm GenerateFacet(\( \hat{\alpha}_t^n, \tilde{\Gamma}_{t+1} \)) (Proposition 3).

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

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

\[\alpha_t \cdot b_t^n \leq \bar{\alpha}_t(b_t^n) \cdot \hat{b}_t^n \quad i = 0, ..., n - 1\]
\[\alpha_t \cdot b_t^n = \bar{\alpha}_t(\hat{b}_t^n) \cdot \hat{b}_t^n \quad j = 1, ..., |S| \]

Let \( \beta_1, ..., \beta_{|S|} \in [0, 1] \) such that \( \hat{b}_t^n = \sum_{j=1,...,|S|} \beta_j b_t^n \) and \( \sum_{j=1,...,|S|} \beta_j = 1 \). Such \( \beta_j \) and \( b_t^n \) always exist if we assume that there are at least \( |S| \) beliefs \( b_t^n \) affinely independent (for example, by setting the \( |S| \) first \( b_t^n \) to the extreme points of the simplex \( B \)). \( \bar{\alpha}_t \cdot b_t^n \) can be easily re-written
\[\sum_{j=1,...,|S|} \beta_j(\bar{\alpha}_t(b_t^n) \cdot b_t^n) \]

This technique of linearization is similar to considering interpolations of \( BL(\hat{V}_{t+1}) \) 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 \( b_0^n, ..., b_{n-1} \) be \( n \geq |S| \) beliefs. Problem \( QP_n(b_0^n, ..., b_{n-1}) \) can be reformulated as the following MILP, called \( MILP_n (b_0^n, ..., b_{n-1}) \), containing only continuous variables except \( n \) that are 0-1 variables.

\[
\begin{align*}
\max & g_t \\
\text{s.t.} & g_t \leq W_t - U_t \\
& W_t = \sum_{i=1,...,n} \beta_i(\bar{\alpha}_t(b_t^n) \cdot \hat{b}_t^n) \\
& \sum_{j=1,...,|S|} \beta_j = 1 \\
& b_t^n \cdot 1 = 1 \\
& b_t^n = \sum_{i=1,...,n} \beta_i \hat{b}_t^n \\
& \alpha_t \cdot b_t^n + y_t = \bar{\alpha}_t(\hat{b}_t^n) \cdot \hat{b}_t^n \quad i = 0, ..., n - 1 \\
& y_t \leq M(1 - x_t) \quad i = 0, ..., n - 1 \\
& U_t \geq \bar{\alpha}_t \cdot b_t^n \quad \tilde{\alpha}_t \in \tilde{\Gamma}_t \\
& \beta_i \leq x_t, i = 0, ..., n - 1 \\
& \sum_{i=0,...,n-1} x_i \leq |S| \\
& b_t^n \geq 0 \\
& y_t \geq 0, i = 0, ..., n - 1 \\
& \beta_i \leq 1, i = 0, ..., n - 1 \\
& \beta_i \geq 0, i = 0, ..., n - 1 \\
& x_t \in \{0, 1\}, i = 0, ..., n - 1 \\
& W_t, U_t, g_t \geq 0
\end{align*}
\]

\( MILP_n \)

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 \( \alpha_t \cdot b_t^n \).

Any optimal solution of \( MILP_n \) is also an optimal solution of \( QP_n \). We keep the same notation for the solutions: \( (\hat{\alpha}_t^n, \hat{\beta}_t^n, \tilde{\alpha}_t^n, \tilde{\gamma}_t^n) \).

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 \( t \), \( \hat{V}_t, \hat{V}_{t+1} \) and a representation \( \bar{B}_t \) of the belief space \( B \), one can solve approximately the optimization problem corresponding to Equation 6 to within specified precision \( c_p \) using Algorithm 1, called FBB (Find Best Belief).
Algorithm 1 Find the best belief for expanding $B_t$ according to Equation 6 to within specified precision $\epsilon_p$

1: procedure FBB($t, B_t, \Gamma_t, \Gamma_{t+1}, \epsilon_p$)  
2: \hspace{10pt} $n \leftarrow 0$  
3: \hspace{10pt} for $s \in S$ do  
4: \hspace{20pt} $b_t^s \leftarrow e_s$  
5: \hspace{10pt} $n \leftarrow n + 1$  
6: \hspace{10pt} while $\Delta > \epsilon_p$ do  
7: \hspace{20pt} $(\alpha_t^n, \beta_t^n, \gamma_t^n) \leftarrow$ Solve MILP$_n(b_t^0, ..., b_t^{n-1})$  
8: \hspace{20pt} $\Delta \leftarrow \beta_t^n = \alpha_t^n - \sum_{b_t} (\alpha_t^n \cdot b_t^n)$  
9: \hspace{20pt} $F_t^n \leftarrow \text{GenerateFacet}(\alpha_t^n, \Gamma_{t+1})$  
10: \hspace{20pt} $b_t^n \leftarrow b_t^n$  
11: \hspace{10pt} $n \leftarrow n + 1$  
12: return $\gamma_t^n, b_t^n$

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 $\epsilon$, using Algorithm 2 by setting $\epsilon_p \leq \epsilon$.

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

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

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

Algorithm 3 $\alpha$-min: Solve POMDP with a maximum number $N$ of $\alpha$-vectors, to within precision $\epsilon_p$.

1: procedure $\alpha$-MIN($H, A, S, O, N, \epsilon, \epsilon_p$)  
2: \hspace{10pt} $\tilde{P}_{T-1} \leftarrow \{r_a | a \in A, r_a$ is not dominated}  
3: \hspace{10pt} gap$_{T-1} = 0$  
4: \hspace{10pt} for $t \in H$ do  
5: \hspace{20pt} $B_t = b_t^{\text{init}}$ \hspace{5pt} $\triangleright$ $b_t^{\text{init}}$ is an arbitrary belief  
6: \hspace{20pt} gap$ \leftarrow \infty$  
7: \hspace{20pt} for $t = T-2, T-3, ..., 0$ do  
8: \hspace{30pt} while gap$ \geq \frac{\epsilon}{T-t}$ do  
9: \hspace{40pt} $(\text{gap}_t, \beta_t) \leftarrow \text{FBB}(t, B_t, \tilde{P}_t, \Gamma_t, \tilde{P}_{t+1}, \epsilon_p)$  
10: \hspace{40pt} $B_t \leftarrow B_t \cup \{\beta_t\}$  
11: \hspace{40pt} $\Gamma_t \leftarrow \Gamma_t \cup (\alpha_t(\beta_t))$  
12: \hspace{30pt} gap$ \leftarrow \sum_{t \in H^t} \text{gap}_t$  
13: return $\{\Gamma_t, t \in H\}$

5 Experiments

We first assess the performance of $\alpha$-min on four small finite-horizon POMDP problems from the literature 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 $\gamma = 1$. $\alpha$-min results were obtained using a fixed number of $\alpha$-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 $\alpha$-min is encouraging with surprisingly good gaps obtained considering the small quantity of $\alpha$-vectors (Table 1).

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

Finally, we illustrate the benefits of using $\alpha$-min to best manage or monitor four sub-populations of Sumatran tigers with...
Table 2: Improvement of the lower bound and the gap as the number of \( \alpha \)-vectors is increased.

| \( \alpha \)-problem | algorithm | \(|\alpha|\) | LB \( T=10 \) | gap \( T=10 \) |
|----------------------|-----------|--------------|----------------|----------------|
| milos-aaai97         | \( \alpha \)-min | 6            | 44.96          | 122.78         |
| milos-aaai97         | \( \alpha \)-min | 10           | 46.67           | 115.50         |
| milos-aaai97         | \( \alpha \)-min | 15           | 49.75           | 104.68         |
| milos-aaai97         | \( \alpha \)-min | 20           | 50.31           | 104.62         |
| milos-aaai97         | \( \alpha \)-min | 30           | 50.73           | 80.40          |

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 \( \alpha \)-min. In this case, the solution is guaranteed to be at most 10% from the optimal strategy (\( \text{gap} < \frac{L_B}{10} \)). 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 Sarpsop solution.

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.

\( \alpha \)-min greedily expands the set of beliefs and the set of \( \alpha \)-vectors by adding iteratively new beliefs and \( \alpha \)-vector which aim to reduce the current gap as much as possible, until \( N \) \( \alpha \)-vectors have been added. One interesting possible future direction would be to improve \( \alpha \)-min in order to generate a set of \( \alpha \)-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 \( b_0 \). Taking advantage of \( b_0 \) 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].

Table 3: Main notations table.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_t )</td>
<td>Lower bound of the value function ( V_t )</td>
</tr>
<tr>
<td>( \bar{\Gamma}_t )</td>
<td>Set of ( \alpha )-vectors describing ( \bar{V}_t )</td>
</tr>
<tr>
<td>( \bar{\alpha}_t )</td>
<td>( \alpha )-vector belonging to ( \bar{\Gamma}_t )</td>
</tr>
<tr>
<td>BL</td>
<td>Bellman operator</td>
</tr>
<tr>
<td>( \bar{\alpha}_t )</td>
<td>Such that ( \forall b_t \in B, \bar{\alpha}<em>t(b_t) \cdot b_t = BL(\bar{V}</em>{t+1})(b_t) )</td>
</tr>
</tbody>
</table>

References


\(^3\)https://sites.google.com/site/ijcaialphamin/home


