Symbolic Dynamic Programming for Continuous State and Observation POMDPs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Partially-observable Markov decision processes (POMDPs) provide a powerful model for real-world sequential decision-making problems. In recent years, point-based value iteration methods have proven to be extremely effective techniques for finding (approximately) optimal dynamic programming solutions to POMDPs when an initial set of belief states is known. However, no point-based work has provided exact point-based backups for both continuous state and observation spaces, which we tackle in this paper. Our key insight is that while there may be an infinite number of possible observations, there are only a finite number of observation partitionings that are relevant for optimal decision-making when a finite, fixed set of reachable belief states is known. To this end, we make two important contributions: (1) we show how previous exact symbolic dynamic programming solutions for continuous state MDPs can be generalized to continuous state POMDPs with discrete observations, and (2) we show how this solution can be further extended via recently developed symbolic methods to continuous state and observations to *derive* the minimal relevant observation partitioning for potentially correlated, multivariate observation spaces. We demonstrate proof-of-concept results on uni- and multi-variate state and observation steam plant control.

1 Introduction and Related Work

Partially-observable Markov decision processes (POMDPs) are a powerful modeling formalism for real-world sequential decision-making problems [3]. In recent years, point-based value iteration methods [5, 10, 11] have proved extremely successful at scaling (approximately) optimal POMDP solutions to large state spaces when a set of initial belief states is known.

More recent work has extended point-based methods to both continuous state and continuous observation spaces, but no work has tackled both jointly without sampling. [6] provides exact point-based backups for continuous state and discrete observation problems (with approximate sample-based extensions to continuous actions and observations), while [2] provides exact point-based backups for discrete state and continuous observation problems (where multivariate observations must be independent). While restricted to discrete state, [2] provides an important insight that we exploit in this work: *only a finite number of partitionings of the observation space are required to distinguish between the optimal conditional policy over a finite set of belief states*.

Our contributions in this work are two-fold. First, we extend symbolic dynamic programming for continuous state MDPs [8] to the case of continuous state and discrete observation POMDPs. This provides an expressive and concrete instantiation of the framework in [6] (which only abstractly required that integrals were computable) in that it ensures the closed-form of all α-functions for *all horizons* is a symbolic piecewise case statement, even for POMDPs with *arbitrary* continuous rewards and transitions with discrete noise (i.e., a finite mixture of deterministic transitions). Second,

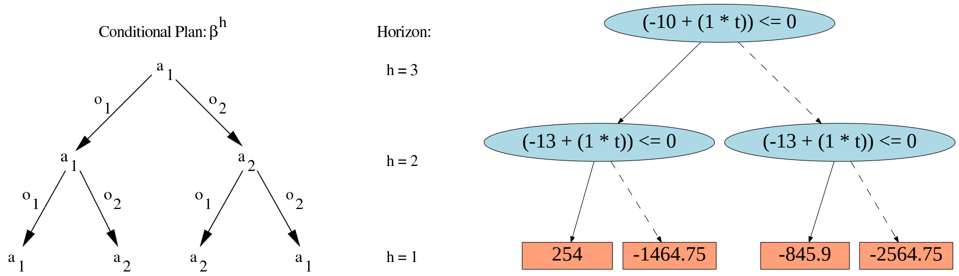

Figure 1: Left: Example conditional plan β^h for discrete observations. Right: The maximum α vector of iteration 4 for the Power Plant as a decision diagram: the *true* branch is solid, the *false* branch is dashed.

we extend this symbolic dynamic programming algorithm to the case of continuous observations (while restricting the transition and reward functions to piecewise polynomials) and extend the work of [2] to *derive* relevant observation partitions for potentially correlated multivariate continuous observation spaces by exploiting the piecewise polynomial integration operation of [7] and the multivariate symbolic maximization technique of [8]. We demonstrate proof-of-concept results on uni- and multi-variate state and observation steam plant control showing the tractability of our exact symbolic dynamic programming point-based approach.

2 DC-POMDP Model

We assume familiarity with MDPs and introduce discrete and continuous partially observable MDPs (DC-POMDPs) as an extension to their DC-MDP subset [8]. A DC-POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \gamma, h \rangle$. States $\mathcal{S}$ are represented by vectors of $(\vec{d_s}, \vec{x_s}) = (d_{s_1}, \ldots, d_{s_n}, x_{s_1}, \ldots, x_{s_m})$ where each $d_{s_i} \in \{0,1\}$ $(1 \leq i \leq n)$ is a discrete boolean and each $x_{s_j} \in \mathbb{R}$ $(1 \leq j \leq m)$ is a continuous variable. We assume a finite, discrete action space $\mathcal{A} = \{a_1, \ldots, a_p\}$. Observations $\mathcal{O}$ are given by the vector $(\vec{d_o}, \vec{x_o}) = (d_{o_1}, \ldots, d_{o_p}, x_{o_1}, \ldots, x_{o_q})$ where each $d_{o_i} \in \{0,1\}$ $(1 \leq i \leq p)$ is boolean and each $x_{o_j} \in \mathbb{R}$ $(1 \leq j \leq q)$ is continuous.

Three functions are required for modeling DC-POMDPs: (1) $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ a Markovian transition model defined as the probability of the next state given the action and previous state; (2) $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ a reward function which returns the immediate reward of taking an action in some state; and (3) an observation function defined as $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow [0,1]$ which gives the probability of an observation given the outcome of a state after executing an action. A discount factor $\gamma,\ 0 \leq \gamma \leq 1$ is used to discount rewards t time steps into the future by γ^t.

We use a dynamic Bayesian network (DBN) to compactly represent the transition model over the factored state variables and we likewise use a Bayesian network to represent the observation model:

$$\mathcal{T}:\ P(\vec{d_s}', \vec{x_s}' | \vec{d_s}, \vec{x_s}, a) = \prod_{i=1}^{n} P(d'_{s_i} | \vec{d_s}, \vec{x_s}, a) \prod_{j=1}^{m} P(x'_{s_j} | \vec{d_s}, \vec{d_s}', \vec{x_s}, a).$$

$$\mathcal{Z}:\ P(\vec{d_o}, \vec{x_o} | \vec{d_s}, \vec{x_s}, a) = \prod_{i=1}^{n} P(d_{o_i} | d_{s_i}, a) \prod_{j=1}^{m} P(x_{o_j} | x_{s_j}, a).$$

For simplicity of algorithmic exposition, we do not allow synchronic arcs in the transition representation, but note that to incorporate such arcs only requires restrictions on the variable elimination ordering during dynamic programming. In these Bayes nets, probabilities over *discrete* variables are represented using conditional probability functions (CPF) in the form of $P(d'_i | \vec{d}, \vec{x}, a)$. Probabilities for *continuous* variables $P(x'_j | \vec{d}, \vec{d'}, \vec{x}, a)$ are represented by *piecewise functions* (PWFs) that are first-order Markov and deterministic, i.e. the probabilities are encoded using the Dirac $\delta[\cdot]$ function. The reward function $R(\vec{d}, \vec{x}, a)$ is set to be any general piecewise function. We next provide an intuitive example to make this DC-POMDP framework more concrete.

Example (Power Plant) [1] *The steam generation system of a power plant aims to provide hot steam to the steam turbine which in turn produces electricity. Feed-water is exposed to heat in the tubes leading to the water tank, where water evaporation is performed under specific pressure and* *temperature. A reward is obtained when electricity is generated through steam production and the pressure and temperature are within safe ranges. Mixing water and steam makes the respective pressure and temperature observations $p_o \in \mathbb{R}$ and $t_o \in \mathbb{R}$ on the underlying state $p_s \in \mathbb{R}$ and $t_s \in \mathbb{R}$ highly uncertain. Action choices are either to open or close a presure valve $a \in \{open, close\}$.*

For the POWER PLANT example, the transition and reward functions are defined as below:

$$P(t'_s|\vec{t_s}, a) = \delta\left[t'_s - \begin{cases}(a = open) & : t_s - 5\\ (a \neq open) & : t_s + 7\end{cases}\right] \qquad R(t_s, a) = \begin{cases}(a = open) & : -1\\ (a \neq open) \wedge (t_s > T) & : -1000\\ (a \neq open) \wedge \neg(t_s > T) & : 100\end{cases}$$

This example will try to maximize its reward using observations on the temperature. The observations received before closing the plant to a safe temperature are therefore critical. For the observation model we begin with a discrete model for the observation z with values $high, low$ such as the following for the $open$ action:

$$P(z = high|t'_s, open) = \begin{cases}t_s <= 15 & : 0.9\\ \neg(t_s <= 15) & : 0.1\end{cases} \qquad P(z = low|t'_s, open) = \begin{cases}t_s <= 15 & : 0.1\\ \neg(t_s <= 15) & : 0.9\end{cases}$$

After defining the solution to the discrete observation with a continuous state DC-POMDP, we can extend this to consider continuous observations using a continuous uniform noise distribution which is more realistic in many domains.

$$P(t_o|t'_s, open) = \begin{cases}(t_o > t_s - 5) \wedge (t_o < t_s + 5) & : 0.1\\ \neg((t_o > t_s - 5) \wedge (t_o < t_s + 5)) & : 0\end{cases}$$

Next we define the general solution to solving DC-POMDPs.

3 DC-POMDP solution

The DC-POMDP will choose actions in order to compute an optimal plan over time. This plan is the policy π of the agent depending on the history of actions and observations over a Markovian state. In the partially observable case where there is uncertainty on the states, the policy is a function over a continuous set of probability distributions over $|\mathcal{S}|$ known as the belief state $b = P(x_s, d_s)$.

At any of the horizon steps, a DC-POMDP policy $\pi(b)$ is defined as the iterative plan on executing an action and observing all probable outcomes of the set of observations from the $(h - 1)$-step. The belief state is updated after this according to the Bayes rule: $\sum_{d_s} \int_{x_s} b(xd_s)T(xd'_s, xd_s, a)Z(xd'_s, xd_o, a)$. [1] The optimal policy is defined over the belief state as the sum of the expected discounted rewards over horizon h starting with belief b_0.

$$V^h_{\Pi^*}(\vec{b}) = E_{\pi^*}\left[\sum_{h=0}^{H} \gamma^t \cdot r_h \Big| \vec{b}_0 = \vec{b}\right]$$

Solving the value function for the optimal policy is performed using Dynamic programming algorithms such as value iteration. The possible set of conditional plans β^h correspond to a decision tree. Figure 1 left, shows the decision tree for the simple case of three horizons with discrete observations. It represents a conditional plan consisting of an action and two possible observation strategies. For continuous observations, conditional plans are decision trees with infinitely many branches. In our solution we show that instead of a policy tree with infinite branches, we can create partitions based on relevant observation values, resulting in a policy tree similar to that of figure 1. For discrete observations and discrete states the number of belief states is infinite, but the optimal value function for h is proven to be a piecewise linear and convex function of the belief state [9]. The optimal value function is the maximum value over a finite set of alpha vectors α_i^h:

$$V^h_{\Pi^*}(\vec{b}) = \max_{\alpha_i^h \in \Gamma^h} \alpha_i^h \cdot \vec{b} \tag{1}$$

where the belief states $\vec{b}$ and each $\alpha_i^h \in \Gamma^h$ are of dimension $|\mathcal{S}|$.

[1] For simiplicity we use $(xd_s = x_s, d_s)$ and $(xd_o = x_o, d_o)$

Algorithm 1: PBVI(DC-POMDP, H,$ContObs$, b_i) $\longrightarrow (V^h, \pi^{*,h})$

```
1  begin
2      $V^0 := 0, h := 0$
3      while $h < H$ do
4          $h := h + 1, \Gamma^h := \emptyset$
5          foreach $a \in A$ do
6              $\Gamma^h_a := \emptyset$
7              if $ContObs = true$ then
8                  $(O^h, P(xd_{o_i}|xd_{s_i}))$ := GenRelObs($\Gamma^{h-1}, a, b_i$)
9              foreach $xd_{o_i} \in$ O$^h$ do
10                 foreach $\alpha_j^{h-1} \in \Gamma^{h-1}$ do
11                     $\alpha_j^{h'}$ = Prime($\alpha_j^h$) // $\forall\ d_i \to d_i'$ and $\forall x_i \to x_i'$
12                     $\Gamma^h_{a,xd_{o_i},j}$ := $P(xd_{o_i}|xd_{s_i}) \otimes$ Backup($\alpha_j^{h'}, a$)
13                 $\Gamma^h_{a,xd_o}$ := $\boxplus \Gamma^h_{a,xd_{o_i}}$
14             $\Gamma^h_a$ := $R_a \oplus \gamma \cdot \Gamma^h_{a,xd_o}$
15             $\Gamma^h$ := $\Gamma^h \cup \Gamma^h_a$
16         $\Gamma^h$ := Prune($\Gamma^h$)
17         $V^h$ := $\max_{\alpha_j \in \Gamma^h} b_i \cdot \alpha_j$
18         $\pi^{*,h}$ := $\arg\max_a \Gamma^h_a$
19         if $V^h = V^{h-1}$ then
20             break // Terminate if early convergence
21
22     return $(V^h, \pi^{*,h})$
23 end
```

Point-based dynamic programming backups compute the optimal value function given a belief state b. The best conditional plan for the given belief, corresponds to an α-vector which is computed using point-based backups:

$$\alpha^{h+1}(b) = R(b) + \gamma \int_{xd_o} P(xd_o|a, b)\alpha^{h+1}(b)$$

where $P(xd_o|a, b) = \int_{xd_s} \int_{xd'_s} b(s)P(xd'_s|xd_s, a)P(xd_o|xd'_s, a)$.

Calculating all possible ways of building the value function independent of a certain belief state can also define the optimal value function. To derive the Γ-set of all possible α-vectors for each horizon, we use the value iteration algorithm from [4]. Initializing $\alpha_1^0 = \vec{0}$ and $\Gamma^0 = \{\alpha_1^0\}$, Γ^h is obtained from Γ^{h-1} using the backup operation defined in (2):[2]

$$g^h_{a,o,j} = \int_{x_{s'}} \sum_{d_{s'}} p(xd_o|xd'_s, a)p(xd'_s|xd_s, a)\alpha^h_j(xd'_s) \tag{2}$$

$$\Gamma^h_a = r_a + \gamma \boxplus_{o \in \mathcal{O}} \left\{ g^h_{a,o,j}(\cdot) \right\}_j ; \forall \alpha_j^{h-1} \in \Gamma^{h-1} \tag{3}$$

$$\Gamma^h = \bigcup_a \Gamma^h_a \tag{4}$$

where $r_a = r(x_s, d_s, a)$. Lines 7–15 of the PBVI algorithm show the above in a general algorithm used for both discrete and continuous observations. We will discuss this according to our symbolic framework in the next section. For all α-vectors in the current Γ-set the PRUNE function uses an LP-solver to prune any inconsistency in the branches of this vector in line 16 similar to the approach in [**?**]. A pairwise dominance test is also performed where for all j, k: $\alpha_j(\vec{s}) \geq \alpha_k(\vec{s})$ The result of this pairwise test is a dominant α-vector which is added to the set of Γ^h.

[2] The $\boxplus$ of sets is defined as $\boxplus_{j \in \{1,\ldots,n\}} S_j = S_1 \boxplus \cdots \boxplus S_n$ where the pairwise cross-sum $P \boxplus Q = \{\vec{p} + \vec{q} | \vec{p} \in P, \vec{q} \in Q\}$.

Note that the α-vector set grows exponentially in the number of observations, i.e., $|\Gamma^h| = |\mathcal{A}||\Gamma^{h-1}|^{|\mathcal{O}|}$ during the cross-sum operation. For this reason even for a small number of discrete observations, tractable solutions may be hard to define. Here we do not restrict our observations to discrete ones, but extend it to a fully continuous approach where lines 7–8 will be executed to generate relevant observation partitions based on a belief state b_i. We use the number of observation partitions $|\mathcal{O}|$ as the branching factor in the policy tree. We present the symbolic approach required to partition this continuous space.

4 Symbolic Dynamic Programming

Dynamic programming provides exact solutions to POMDP problems for small domains. As the domain grows, fully enumerating the state, action and observation space leads to intractable solutions even for low horizons. Avoiding this enumeration often requires approximating the solutions. We claim an exact solution using symbolic dynamic programming. The SDP solution for fully observable domains has been presented in [8] Our symbolic algorithm deals with continuous observations and states by generating the relevant observation partition at every time step and performing the backup operation using α-vectors. We define the requirements to implement the value iteration algorithm for DC-POMDP.

4.1 Case Representation and Extended ADDs

The POWER PLANT example represented all functions using the case structure which can be generally defined as:

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases}$$

Where ϕ_i are disjoint logical formulae defined over the state $(\vec{d}, \vec{x})$ with logical $(\wedge, \vee, \neg)$ combinations of boolean variables and inequalities $(\geq, >, \leq, <)$ over continuous variables. The f_i are function definitions either linear or quadratic in the continuous parameters.

For *unary operations* such as scalar multiplication $c \cdot f$ (for some constant $c \in \mathbb{R}$) or negation $-f$ on case statements is simply to apply the operation on each case partition f_i $(1 \leq i \leq k)$. A *binary operation* on two case statements, takes the cross-product of the logical partitions of each case statement and performs the corresponding operation on the resulting paired partitions. The cross-sum $\oplus$ of two cases is defined as the following:

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \begin{cases} \phi_1 \wedge \psi_1 : & f_1 + g_1 \\ \phi_1 \wedge \psi_2 : & f_1 + g_2 \\ \phi_2 \wedge \psi_1 : & f_2 + g_1 \\ \phi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

Likewise $\ominus$ and $\otimes$ are defined by subtracting or multiplying partition values. Inconsistent partitions can be discarded when they are irrelevant to the function value. The next three operations are required in defining the value iteration algorithm. A *symbolic case maximization* is defined as below:

$$\max\left(\begin{cases} \phi_1 : f_1 \\ \phi_2 : f_2 \end{cases}, \begin{cases} \psi_1 : g_1 \\ \psi_2 : g_2 \end{cases} \right) = \begin{cases} \phi_1 \wedge \psi_1 \wedge f_1 > g_1 : f_1 \\ \phi_1 \wedge \psi_1 \wedge f_1 \leq g_1 : g_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 > g_2 : f_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 \leq g_2 : g_2 \\ \vdots \qquad\qquad\qquad \vdots \end{cases}$$

We also require the following SDP operations on case statements already introduced in the literature:

- Restriction $f|_\phi$: Takes a function f to restrict only in cases that satisfy some formula ϕ as defined in [8].

- Substitution $f\sigma$: Takes a set σ of variables and their substitutions, where the LHS of the substitution operator / represents the substitution variable and the RHS of the / represents the expression that should be substituted in its place as in [8].
- Integration $\int_{x_1} f dx_1$: Eliminates continuous variables by taking the integral of the polynomial function f with respect to linear constraints on x_1 [7].

The data structure of *extended ADD* (XADD) [8] is used to support case statements and the required operations. Here we use continuous SDP for both states and observations in DC-POMDPs. The α-vectors partition into states of equal value, both of which can be represented using XADDs. For increasing efficiency linear constraint feasibility checkers such as LP solvers allow pruning unreachable paths in the XADD. Figure 1 (right) is an example of an XADD representation.

4.2 PBVI for DC State and Discrete Observations

For discrete and continuous states and discrete observations, we use the symbolic version of Monahan's algorithm as the DC-POMDP solution represented in PBVI. The same approach is required for continuous observations only based on point-based dynamic programming backups.

Given an initial set of beliefs, we can compute the relevant observation partitions explained in the next section and then continue in a similar fashion of having discrete observations. Note that if the input to this algorithm is continuous observations $ContObs = true$ then line 8 finds the relevant observation partitions, else discrete observations are assumed and their probabilities are given in the problem specifications such as section 2 of this paper. Also a belief is given as the input and often several beliefs are required to cover the belief space and thus the algorithm is executed for each one of $b_i \in \mathbb{B}$.

For each iteration, lines 11–12 product the following equation:

$$g^h_{a,o,j} = \int_{x_{s'}} \sum_{d_{s'}} p(xd_o|xd'_s,a)p(xd'_s|xd_s,a)\alpha^h_j(xd'_s)$$

where the α-vector set are case functions primed by substitution in Prime and Backup performs regression on the α-vectors. Boolean regression is performing a symbolic restriction while a continuous regression is a symbolic substitution as defined previously. Line 13 takes the cross-sum of the two sets which takes every possible combination of observations according to the α-vectors. All vectors are then multiplied by the discount γ and added with the reward in line 14 where the addition is an augmented $\oplus$ for adding each vector with the reward function. The same operation is performed for all actions and then in line 15 the Γ set for each action are added to the set of Γ^h to build all the vectors for the next horizon. The α-vector that maximizes the value-function with respect to this belief and save it as the new Γ-set for the next horizon. The action from which this α-vector was chosen from is the maximum policy. This operation is performed for each of the belief points $b_i \in B$ and for each belief point one dominant α-vector is stored in the Γ-set. The algorithm iterates until convergence. In the next section we discuss the main part of our algorithm which is generating relevant observation partitions.

4.3 PBVI for DC State and DC Observations

The main operations required for value iteration in DC-POMDP is generating relevant observations to perform the bellman backup operation. The GenRelObs algorithm defines the steps required to generate these partitions. We use the second iteration of the Power Plant example to demonstrate partitioning of the entire continuous observation space. The main assumption is to consider some belief state and obtaining the relevant partitions according to that belief. We choose these beliefs such that they cover the entire probable state space; e.g. $b_1 : U[t_s; 2, 6]$ and $b_2 : U[t_s; 6, 11]$.

Starting with an empty set of relevant observation partitions z, the algorithm first eliminates the next state variable xd'_s. It multiplies the observation model and the transition model in the set of α-vectors of the previous horizon: $\alpha_j(xd_s, xd_o) := \int_{s'} P(xd_{o_i}|xd'_{s_i}, a) * P(xd'_{s_i}|xd_{s_i}, a) * \alpha_j$. This operation of line 3 in GenRelObs is similar to that of Regress multiplied by the observation regressed model. For continuous variables the continuous regression and for discrete variables boolean restriction is used as defined in the previous section. Assuming that after $h = 1$

Algorithm 2: `GenRelObs`$(\Gamma^h, a, b_i) \longrightarrow P(z)$

```
1  begin
2      $z := 0$; for all $\alpha_j$ in $\Gamma^{h-1}$ do
3          $\alpha_j(xd_s, xd_o) := \int_{s'} P(xd_{o_i}|xd'_{s_i}, a) \cdot P(xd'_{s_i}|xd_{s_i}, a) \cdot \alpha_j$;
4      for all $j$ in $\alpha_j(xd_s, xd_o)$ do
5          $\delta_j(xd_o) := \int_{xd_{s_i}} b_i \cdot \alpha_j(xd_s, xd_o)\}$
6      for all $j$ in $\delta_j(xd_o)$ do
7          $z := max(\delta_j(xd_o), z)$
8      // $\phi_{z_k}$ logical term on each partition of $z$
9      $P(z_k) := \int_{xd_o} \int_{xd_s} P(xd_o|xd'_s, a) * P(xd'_s|xd_s, a) * b_i * \mathbb{I}[\phi_{z_k}] d_{xd_o} d_{xd_s}$
10     return $P(z)$
11 end
```

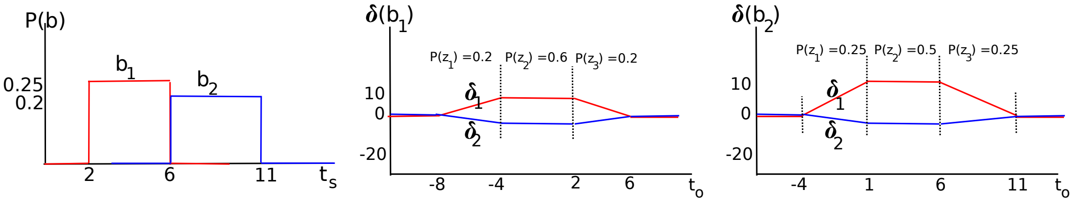

Figure 2: *(left)* Beliefs b_1, b_2 for 1D `Power Plant` example; *(middle)* Observation dependent function for b_1 that partition the observation space into 3 regions with different probabilities $P(z_i)$; *(right)* Relevant observation partitions and their probabilities for b_2.

the set of α-vectors are defined as:

$$\alpha_1^1(s) = \begin{cases} (t_s > 200) & : -1000 \\ \neg(t_s > 200) & : 100 \end{cases} \qquad \alpha_2^1(s) = \top : -1$$

For $h = 2$ the resulting α-vectors after this regression step are functions of the observation and current state:

$$\alpha_1^2(s,o) = \begin{cases} (t_s < 15) \wedge (t_s - 10 < t_o < t_s) & : 10 \\ (t_s > 15) \wedge (t_s - 10 < t_o < t_s) & : -100 \\ \neg(t_s - 10 < t_o < t_s) & : 0 \end{cases} \quad \alpha_2^2(s,o) \quad = \begin{cases} (t_s - 10 < t_o < t_s) & : -0.1 \\ \neg(t_s - 10 < t_o < t_s) & : 0 \end{cases}$$

We need the α-vectors to partition on the observation space, thus next we eliminate the current state variable in line 4–5. During this integration the current state variable is factored out by multiplying the belief in each of the current α-vectors. The resulting δ-functions only depend on the observation: $\delta_j(xd_o) := \int_{xd_{s_i}} b_i * \alpha_j(xd_s, xd_o)$. For $b_1 : U[t_s; 2, 6]$ the two vectors above the resulting δ-functions are defined below:

$$\delta_1(o) = \begin{cases} (2 < t_o < 6) & : 15 - 2.5 * t_o \\ (-4 < t_o < 2) & : 10 \\ (-8 < t_o < -4) & : 20 + 2.5 * t_o \end{cases} \qquad \delta_2(o) \quad = \begin{cases} (2 < t_o < 6) & : -0.15 + 0.025 * t_o \\ (-4 < t_o < 2) & : -0.1 \\ (-8 < t_o < -4) & : -0.2 - 0.025 * t_o \end{cases}$$

The observation dependent δ-functions divide the observation space into regions which can yield the optimal policy according to the belief state b_1. According to continuous observation space defined in [2], for a 1-dimensional observation space with two or more functions of the observation, we need to find the optimal boundaries or partitions of the space. In this work, numerical solutions are proposed to find the roots of any two observation dependent function. Here we use the symbolic power of the max-operator to find all the maximum boundary regions of two or more δ-functions at the same time. For the two δ-functions above the following partitions of the observation space is derived after taking their maximum in line 6–7:

$$\max\Big(\delta_1(o), \delta_1(o)\Big) = \begin{cases} (2 < t_o < 6) & : 15 - 2.5 * t_o \\ (-4 < t_o < 2) & : 10 \\ (-8 < t_o < -4) & : 20 + 2.5 * t_o \end{cases}$$

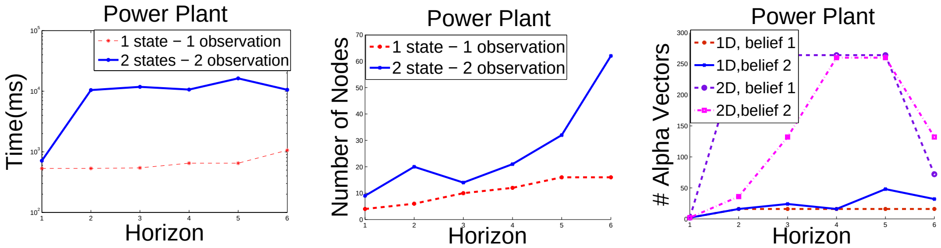

Figure 3: *(left)* Space vs Horizon; *(middle)* Time vs Horizon; *(right)* Number of α-vectors vs Horizon.

The three partitions define 3 observation regions which we can now use in a similar fashion to a discrete observation set if only the probability of each of the three distinct observation partitions is found. If we integrate out the state and observation from the observation model and the belief state regardless of a particular α-vector, it integrates to one: $\int_{xd_o} \int_{xd_s} P(xd_o|xd'_s, a) * P(xd'_s|xd_s, a) * P(s) = 1$ Thus we only need to multiply the indicators of each case partition as as in line 9 in this formula to obtain the probability of each partition such that it sums to one: $P(z_k) := \int_{xd_o} \int_{xd_s} P(xd_o|xd'_s, a) * P(xd'_s|xd_s, a) * b_i * \mathbb{I}[\phi_{z_k}] d_{xd_o} d_{xd_s}$ For our example the following holds:

$$P(z_k) = \begin{cases} (2 < t_o < 6) & : 0.2 \\ (-4 < t_o < 2) & : 0.6 \\ (-8 < t_o < -4) & : 0.1999 \end{cases}$$

Hence we can now use the algorithms and methods of a discrete observation setting using the probabilities of the partitioned observation space in Monahan's algorithm. Next we present some results for 2-Dimensional continuous observation spaces.

5 Empirical Results

We evaluated our continuous POMDP solution using XADDs on the one-dimensional We start by defining the 2-dimensional description of this problem:

2D-POWER PLANT: We consider the more complex model of the power plant similar to [1] where the pressure inside the water tank must be controlled to avoid mixing water into the steam or explosion of the tank. We model the pressure variable p as a partially observable variable from the observation readings of the pressure po. The two actions of increase and decrease are defined based on the change in both the temperature and the pressure. For the increase action the transition functions and the reward function is defined as below:

$$P(p'_s|\vec{p_s}, inc) = \delta \left[p'_s - \begin{cases} (p + 10 > 20) & : 20 \\ \neg(p + 10 > 20) & : p_s + 10 \end{cases} \right] \qquad P(t'_s|\vec{t_s}, inc) = \delta \left[t'_s - (t_s + 10) \right]$$

$$R(t_s, p_s, inc) = \begin{cases} (5 \le p_s \le 15) \wedge (95 \le t_s \le 105) & : 50 \\ (5 \le p_s \le 15) \wedge (t_s \le 95) & : -1 \\ (5 \le p_s \le 15) \wedge (t_s \ge 95) & : -3 \\ (p_s \le 5) & : -3 \\ (p_s \ge 15) & : -5 \end{cases}$$

There is a high reward for staying withing the safe temperature and pressure range since it produces power, else depending on how safe it is to have values higher or lower than the safe range, penalty is defined. As for the decrease action, the transition functions reduce the temperature by 5 units and the pressure by 10 units as long as the pressure stays above zero. For the reward function, we assume that there is always a small penalty for decreasing the values because power can not be generated.

For the observation model we consider two continuous uniform distributions such as the following:

$$P(t_o|t'_s) = \begin{cases} (t_s + 80 < t_o < t_s + 105) & : 0.4 \\ \neg(t_s + 80 < t_o < t_s + 105) & : 0 \end{cases} \qquad P(p_o|p'_s) = \begin{cases} (p_s < p_o < p_s + 10) & : 0.1 \\ \neg(p_s < p_o < p_s + 10) & : 0 \end{cases}$$

We define two rectangular uniform beliefs around the regions of rewarding, so that one needs to increase the values while the other should decrease them: $b_1 : U[t_s; 90, 100] * U[p_s; 0, 10]$ and $b_2 :$

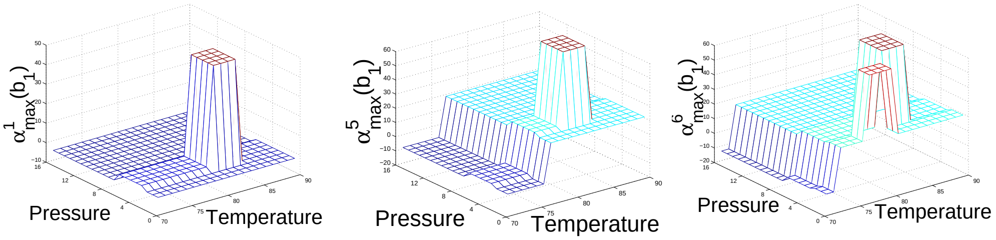

Figure 4: *(left)*Maximum α-vector for b_1 in first iteration; *(middle)* $\alpha^5_{max}(b_1)$; *(right)* $\alpha^5_{max}(b_1)$.

$U[t_s; 90, 130] * U[p_s; 10, 30]$ In Figure 3, a time and space analysis of the two versions of POWER PLANT have been performed for up to 6 horizons. Comparing the two problem sizes demonstrates the effect of the number of state-observation variables in our algorithm. As the algorithm progresses, the time required to compute the probability of the partitions and finding the maximum α-vector with respect to beliefs increases for both problem sizes and significantly more for the 2D version. The stability in time is due to the fact that each stage almost takes the same time since it has converged quickly.Increase in the problem size, increases the partition numbers on the observation space and this produces more α-vectors which also effects the space required to perform the algorithm. The number of vectors stays the same for most horizons and they drop after convergence in the 2D problem instance. The number of nodes in each iteration increases as more space is required to perform higher iterations. This shows that although the 2D instance takes more time and space than the 1D instance, it still converges within reasonable resources.

In Figure 4 we present plots of the maximum α-vectors of belief b_1 for different iterations of the 2D problem instance. Each plot demonstrates the value of each vector as a function of the pressure and temperature. Starting with the first iteration, the value is highest for the reward range $(5 < p < 15 \wedge 95 < t < 105)$ and -1 or less for other places. In the fifth iteration, the value function has partitioned into more pieces, showing how higher temperatures can increase the value without considering the effect of the pressure. In the last plot, horizon 6 has better tuned the value function so that higher temperatures and pressures increase the value of the maximum α-vector and also within the reward range, finer grain partitions have been formed. Also note that the policy depends on the initial belief and for these plots they were based on b_1 which was lower than the reward range. The first action would then mean it has performed an increase while the other iterations decrease and increase respectively.

6 Related Work and Conclusion

This work has used the concept of continuous states and the SDP solution to MDPs from [8] and continuous observations from the work of [2]. It is exclusive in the sense that it brings together the continuous state and observation POMDPs using a symbolic approach. In many applications of the POMDP framework, the state and observation space have been considered as discrete values such as [1] here we avoid this general simplification and work with the real values from sensors.

There has been prior work on approximate solutions to POMDPs with large or continuous state spaces [12], [11] and some has been extended to the continuous observation setting using methods such as sampling [6]. In most approximate continuous state POMDP work, continuous or large discrete actions has been used. Thus we can extend the current work using [13] to contain continuous actions and provide exact or approximate solutions.

We presented the first exact solution to continuous states and observations in a DC-POMDP framework. Transitions and reward functions are defined as piecewise linear functions of the continuous states. Continuous observations require definition of explicit partitions based on the observation model and the belief state of the agent. The key contribution is to define the partitions on the observation space and derive the related probabilities using symbolic integration and maximization. We extend the application to more than one state-observation setting and present the results.

References

[1] Mario Agueda and Pablo Ibarguengoytia. An architecture for planning in uncertain domains. In *Proceedings of the ICTAI 2002 Conference*, Dallas,Texas, 2002.

[2] Jesse Hoey and Pascal Poupart. Solving pomdps with continuous or large discrete observation spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, 2005.

[3] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[4] G. E. Monahan. Survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[5] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Anytime point-based approximations for large pomdps. *J. Artif. Intell. Res. (JAIR)*, 27:335–380, 2006.

[6] J. M. Porta, N. Vlassis, M.T.J. Spaan, and P. Poupart. Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research*, 7:195220, 2006.

[7] Scott Sanner and Ehsan Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *In Proceedings of the 26th AAAI Conference on Artificial Intelligence (***AAAI-12***)*, Toronto, Canada, 2012.

[8] Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes de Barros. Symbolic dynamic programming for discrete and continuous state mdps. In *Proceedings of the 27th Conference on Uncertainty in AI (UAI-2011)*, Barcelona, 2011.

[9] R.D. Smallwood and E.J. Sondik. The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research*, 21(5):107188, 1973.

[10] Trey Smith and Reid G. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[11] M. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for pomdps. *Journal of Articial Intelligence Research (JAIR)*, page 195220, 2005.

[12] S. Thrun. Monte carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1064–1070. MIT Press, 2000.

[13] Zahra Zamani, Scott Sanner, and Cheng Fang. Symbolic dynamic programming for continuous state and action mdps. In *In Proceedings of the 26th AAAI Conference on Artificial Intelligence (***AAAI-12***)*, Toronto, Canada, 2012.