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 our algorithm on a power plant control problem requiring continuous state *and* observations.

1 Introduction

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 [6, 11, 12, 8] 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. [7] 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 conditionally independent). While restricted to discrete states, [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 [10] to the case of continuous state and discrete observation POMDPs. This provides an expressive and concrete instantiation of the framework in [7] (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,

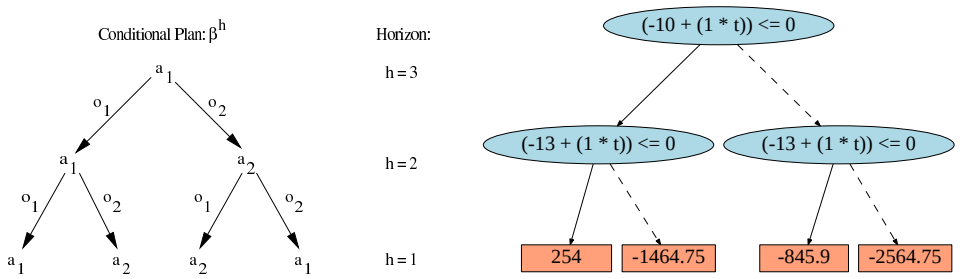

Figure 1: (left) Example conditional plan β^h for discrete observations; (right) an example piecewise α-vector for continuous state t represented as a decision diagram: the *true* branch is solid, the *false* branch is dashed.

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 [9] and the multivariate symbolic maximization technique of [10]. We conclude by demonstrating our algorithm on a power plant control problem requiring bivariate continuous state and observations.

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 [10]. 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 γ, $0 \leq \gamma \leq 1$ 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 transition and observation 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 pressure valve $a \in \{open, close\}$.

We initially present two simple examples labeled **1D-Power Plant** using only the temperature state t_s. First, the transition and reward functions common to both problems are defined as below where simply, the temperate increments or decrements according to the action:

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

Next we introduce the **Discrete Obs. 1D-Power Plant** variant using a discrete binary observation space with possible observations $o \in \mathcal{O} = \{high, low\}$:

$$P(o = high|t'_s, open) = \begin{cases}t'_s <= 15 & : 0.9\\ \neg(t'_s <= 15) & : 0.1\end{cases} \quad P(o = low|t'_s, open) = \begin{cases}t'_s <= 15 & : 0.1\\ \neg(t'_s <= 15) & : 0.9\end{cases} \tag{2}$$

Finally we introduce the **Cont. Obs. 1D-Power Plant** variant with continuous observation t_o. We define the observation probability over t_o as uniform on an interval 10 units wide, centered at t'_s.

$$P(t_o|t'_s, open) = \begin{cases}(t_o > t'_s - 5) \wedge (t_o < t'_s + 5) & : 0.1\\ \neg((t_o > t'_s - 5) \wedge (t_o < t'_s + 5)) & : 0\end{cases} \tag{3}$$

3 DC-POMDP solution

In a DC-POMDP, the agent does not directly observe the states and thus must maintain a belief state $b(xd_s) = P(xd_s)$.[1] For a given belief state $\vec{b} = b(\cdot)$, a POMDP policy π can be represented by a tree corresponding to a conditional plan β. An h-step conditional plan β^h can be defined recursively in terms of $(h-1)$-step conditional plans as shown in Fig. 1.

Our goal is to find a policy π that maximizes the value function, defined as the sum of expected discounted rewards over horizon h starting from initial belief state $\vec{b}$:

$$V^h_\pi(\vec{b}) = E_\pi\left[\sum\nolimits_{t=0}^{h} \gamma^t \cdot r_t \middle| \vec{b}_0 = \vec{b}\right] \tag{4}$$

where r_t is the reward obtained at time t and $\vec{b}_0$ is the belief state at $t = 0$. For finite h and belief state $\vec{b}$, the optimal policy π takes the form of an h-step conditional plan β^h. For $h = \infty$, the optimal discounted ($\gamma < 1$) value function can be approximated arbitrarily closely by using a sufficiently large, but finite h [3].

Even when the state is continuous (but the actions and observations are discrete), the optimal POMDP value function for finite horizon h is a piecewise linear and convex function of the belief state $\vec{b}$ [7] hence V^h is given by a maximum over a finite set of "α-vectors" α^h_i:

$$V^h(\vec{b}) = \max\nolimits_{\alpha^h_i \in \Gamma^h} \alpha^h_i \cdot \vec{b} \tag{5}$$

Later on when we tackle continuous state *and* observations, we note that we will dynamically derive an optimal, finite partitioning of the observation space for a given belief state and hence reduce the continuous observation problem back to a discrete observation problem at every horizon.

The Γ^h in this optimal h-stage-to-go value function can be computed via Monahan's dynamic programming approach to *value iteration* (VI) [4]. Initializing $\alpha^0_1 = \vec{0}$ and $\Gamma^0 = \{\alpha^0_1\}$, Γ^h is obtained from Γ^{h-1} using the backup operation defined in (2):[2]

$$g^h_{a,o,j} = \int_{x_{s'}} \sum_{d_{s'}} p(o|xd'_s, a)p(xd'_s|xd_s, a)\alpha^h_j(xd'_s) \tag{6}$$

$$\Gamma^h_a = r_a + \gamma \boxplus_{o \in \mathcal{O}} \left\{g^h_{a,o,j}(\cdot)\right\}_j ; \forall \alpha^{h-1}_j \in \Gamma^{h-1} \tag{7}$$

[1] For compactness we define $xd_s := (x_s, d_s)$ and $xd_o := (x_o, d_o)$.

[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 $P \boxplus Q = \{\vec{p} + \vec{q} | \vec{p} \in P, \vec{q} \in Q\}$.

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

```
1  begin
2      V^0 := 0, h := 0, Γ^0_PBVI = {α^0_1}
3      while h < H do
4          h := h + 1, Γ^h := ∅, Γ^h_PBVI := ∅
5          foreach b_i ∈ B do
6              foreach a ∈ A do
7                  Γ^h_a := ∅
8                  if ContObs = true then
9                      // Continuous observation set O^h_i for belief b_i
10                     (O^h_i, P(O^h_i|x d_s)) := GenRelObs(Γ^{h-1}_PBVI, a, b_i)
11                 foreach o ∈ O^h_i do
12                     foreach α^{h-1}_j ∈ Γ^{h-1}_PBVI do
13                         α^{h'}_j, := Prime(α^h_j) // ∀ d_i → d'_i and ∀ x_i → x'_i
14                         g^h_{a,o,j} := see Eq (6)
15
16                 Γ^h_a := see Eq (7)
17                 Γ^h := Γ^h ∪ Γ^h_a
18             Γ^h := Prune(Γ^h)
19         foreach b_i ∈ B do
20             α^h_{b_i} := arg max_{α_j ∈ Γ^h} α_j · b_i
21             Γ^h_PBVI := Γ^h_PBVI ∪ α^h_{b_i}
22
23         if Γ^h_PBVI = Γ^{h-1}_PBVI then
24             break // Terminate if early convergence
25
26     return Γ_PBVI
27 end
```

$$\Gamma^h = \bigcup_a \Gamma^h_a \tag{8}$$

where $r_a = r(x_s, d_s, a)$.

Point-based value iteration (PBVI) computes the optimal value function only for a set of belief states $\{b_i\}$ where $b_i := P(xd_s)$. The idea is straightforward and the main modification needed to Monahan's algorithm in Algorithm 1 is the loop from lines 19–21 where only α-vectors optimal at some belief state are retained for subsequent iterations. We note that *if* all beliefs reachable in h steps from some initial set can be finitely enumerated and PBVI is executed for this set of beliefs, then optimality is guaranteed for the PBVI policy over these initial beliefs up to horizon h.

We pause for a moment to make a few notes about the PBVI algorithm in the context of DC-POMDPs. If we have a fixed discrete observation set, i.e., $\forall h\, \mathcal{O}^h = \mathcal{O}$, then the algorithm proceeds as usual, but if we have continuous observation variables then we will need to derive a relevant set of observations on line 10, which we describe in Section 4.3. We need to guarantee that the integral on line 14 can be done in closed-form and we guarantee this for piecewise functions in Section 4.2. Because in general we will deal with α-functions represented as piecewise functions, we only assume that strict dominance testing is implemented for Prune in line 18.

4 Symbolic Dynamic Programming

In this section we take a symbolic dynamic programming (SDP) approach to performing PBVI in Algorithm 1. This requires us to define all of the operations used in PBVI on a closed-form representation. For SDP, this closed-form will be a piecewise case statement and it will be stored compactly as a decision diagram [10]. We first review this form and then proceed to define SDP operations specific to the case of DC-POMDPs.

4.1 Case Representation and Extended ADDs

The **Power Plant** example represented all functions using the case structure which can be generally defined as

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

and this is the form we use to represent all functions in a DC-POMDP. Here 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 general functions, we will restrict them later for the case of continuous observation POMDPs.

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. A *symbolic case maximization* is defined as below:

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

The following SDP operations on case statements require more detail than can be provided here, hence we refer the reader to the relevant literature:

- Restriction $f|_\phi$: Takes a function f to restrict only in cases that satisfy some formula ϕ as defined in [10].
- Substitution $f\sigma$: Takes a set σ of variables and their substitutions (which may be case statements), and carries out all variable substitutions in sequence [10].
- Integration $\int_{x_1} f dx_1$: There are two forms: If x_1 is involved in a δ-function (cf. the transition in Eq. (1)) then the integral is equivalent to a symbolic substitution and can be applied to any case statement [10]. Otherwise, if f is restricted to linear constraints and polynomial values, then the approach of [9] can be applied to yield a result in the same form.

The data structure of the *extended algebraic decision diagram* (XADD) [10] is used to support case statements and the required operations. 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`. In brief we note that all operations as defined in PBVI apply directly to functions represented as case statements. For example, we can rewrite Eq (6) in the following case operation form:

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

Crucially we note that since $p(xd'_s|xd_s, a)$ is always defined using δ-functions (possibly conditioned on stochastic discrete variables) for deterministic transitions as defined in our DC-POMDPs, then the integral can always be computed in closed case form as discussed in Section 4.1. In short, nothing additional is required for PBVI on DC-POMDPs with continuous state and discrete observations — the crucial insight to understand is simply that α-vectors are now represented by case statements and can "grow" with the horizon as they partition the state space more and more finely.

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

1 **begin**
2 $Z := 0$; **for** *all* α_j *in* Γ^{h-1} **do**
3 *// Point-based backup operation*
4 $\alpha_j(xd_s, xd_o) := \int_{x'_s} \bigoplus_{d'_s} P(xd_o|xd'_s, a) \cdot P(xd'_s|xd_s, a) \cdot \alpha_j$
5 **for** *all* j *in* $\alpha_j(xd_s, xd_o)$ **do**
6 *// Generate value of* α*-vector for belief* b_i *as a function of observations*
7 $\delta_j(xd_o) := \int_{x_s} \bigoplus_{d_s} b_i \cdot \alpha_j(xd_s, xd_o)\}$
8 **for** *all* j *in* $\delta_j(xd_o)$ **do**
9 *// Generate partition points on the observation space*
10 $Z := max(\delta_j(xd_o), z)$
11 *//* ϕ_{z_k} *logical term on each partition of* Z
12 *// Generate probabilities and partitions of observations that distinguish optimal* α*-vector (conditional plan)*
13 $P(Z = z_k|b_i) := \int_{x_o} \int_{x_s} \bigoplus_{d_o} \bigoplus_{d_s} P(xd_o|xd'_s, a) * P(xd'_s|xd_s, a) * b_i * \mathbb{I}[\phi_{z_k}] d_{xd_o} d_{xd_s}$
14 **return** $P(z)$
15 **end**

4.3 PBVI for DC State and DC Observations

The additional operation required for PBVI in DC-POMDPs in Algorithm 1 over the discrete observation case is simply *generating relevant observations* as shown on line 10. GenRelObs in Algorithm 2 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 b_i and obtaining the relevant partitions according to that belief. For example, we can choose two beliefs as uniform distributions, e.g. $b_1 : U[t_s; 2, 6]$ and $b_2 : U[t_s; 6, 11]$ as shown in Figure 2 (left). Starting with an empty set of relevant observation partitions $\mathcal{O}^h$, the algorithm first eliminates the next state variable xd'_s. It regresses the the set of α-vectors of the previous horizon back one step via the transition and observation model: $\alpha_j(xd_s, xd_o) := \int_{x'_s} \bigoplus_{x'_d} P(xd_{o_i}|xd'_{s_i}, a) * P(xd'_{s_i}|xd_{s_i}, a) * \alpha_j$. This operation of line 3 in GenRelObs is similar to the point-based backup in (6). For continuous variables the continuous integral and for discrete variables the restriction operation is used as defined in Section 4.1. Assuming that after $h = 1$ the set of α-vectors are defined equal to the reward function of equation (1). For $h = 2$ the resulting α-vectors after the regression step are functions of the observation and current state:

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

We now need the α-vectors as a function of the observation space for a particular belief state, thus next we eliminate the current state variable in line 4–5. The resulting δ-functions now only depend on the observation: $\delta_j(xd_o) := \int_{xd_{s_i}} b_i * \alpha_j(xd_s, xd_o)$. For $b_1 : U[t_s; 2, 6]$ assume the following simple δ-functions:

$$\delta_1(t_o) = \Big\{(-\infty < t_o < \infty) \quad : -0.4 * t_o + 6.8 \qquad \delta_2(t_o) \quad = \Big\{(-\infty < t_o < \infty) \quad : 1.3 * t_o + 3.4$$

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 their work, numerical solutions are proposed to find the roots of any two observation dependent function. Instead, here we leverage the symbolic power of the max-operator defined in Section 4.1 to find all the boundary regions that define for what partitionings of the observation each δ-function is optimal. For the two δ-functions above the following partitions of the observation space is derived after taking their maximum in line 6–7:

$$\max\Big(\delta_1(t_o), \delta_2(t_o)\Big) = \begin{cases} z_1 : (t_o < 2 : -0.4 * t_o + 6.8) \\ z_2 : (t_o > 2 : 1.3 * t_o + 3.4) \end{cases}$$

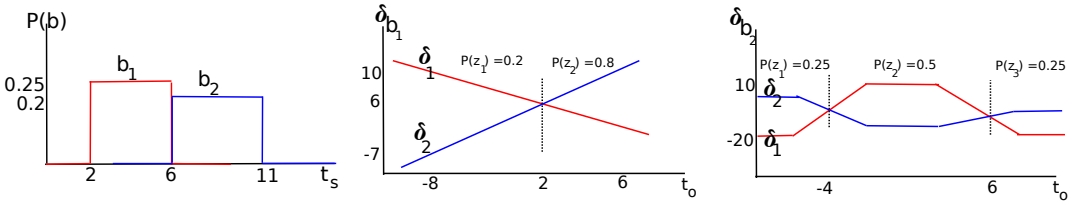

Figure 2: *(left)* Beliefs b_1, b_2 for 1D **Power Plant** example; *(center)* Observation dependent function for b_1 that partition the observation space into 2 regions with different probabilities $P(z_i)$; *(right)* 3 Relevant observation partitions and their probabilities for b_2.

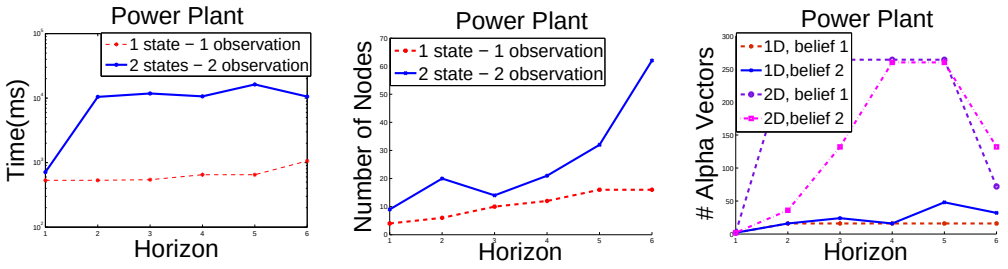

Figure 3: *(left)* Space vs Horizon; *(center)* Time vs Horizon; *(right)* Number of α-vectors vs Horizon.

The two partitions define observation regions which we can now use in a similar fashion to a discrete observation set if only the probability of each of the two distinct observation partitions is found. This is demonstrated visually in Figure 2 (center) for this example and for another example (and belief state) in Figure 2 (right).

Now with the observation partitions derived, all that remains is to calculate probabilities for these relevant observations conditioned on the belief state. Thus we only need to multiply the indicators of each observation partition in this formula to obtain the probability mass lying in each partition:

$$P(z_k|b_i) := \int_{xd_o} \int_{xd_s} P(xd_o|xd'_s, a) * P(xd'_s|xd_s, a) * b_i * \mathbb{I}[\phi_{z_k}] d_{xd_o} d_{xd_s}$$

For our example the probabilities can be calculated as $P(z_1|b_1) = 0.2$ and $P(z_2|b_1) = 0.8$.

Hence we can now use the algorithms and methods of a discrete observation setting using the probabilities of the partitioned observation space in PBVI! Next we present some results for 2-dimensional continuous observation spaces.

5 Empirical Results

We evaluated our continuous POMDP solution using XADDs on the **1D-Power Plant** example and another variant of this problem with two variables, described below.[3]

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 we define:

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

[3]Full problem specifications and Java code to reproduce these experiments are in Google Code (link suppressed) and also in a fully anonymized supplementary archive with instructions, attached to this submission.

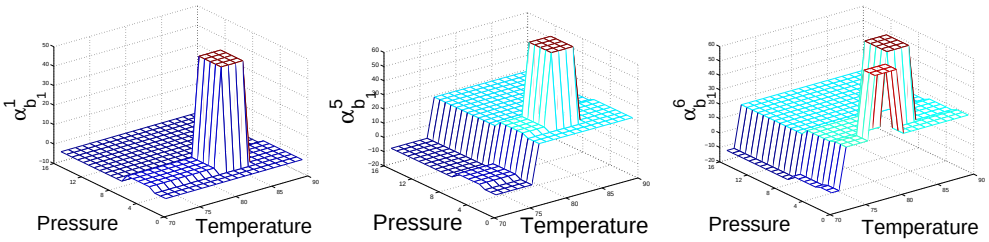

Figure 4: *(left)*Maximum α-vector for b_1 in first iteration; *(center)* $\alpha^5_{max}(b_1)$; *(right)* $\alpha^5_{max}(b_1)$.

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.

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

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 $b_2 : U[t_s; 90, 130] * U[p_s; 10, 30]$ In Figure 3, a time and space analysis of the two versions of **Power Plant** have been performed for up to 6 horizons. 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. 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. 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. 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 $h = 6$ 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.

6 Conclusion

We presented the first exact symbolic operations for `PBVI` in expressive DC-POMDPs with continuous state *and* observations. Unlike related work that has extended to the continuous state and observation setting [7], we do not approach the problem by sampling. Rather, following [2], the key contribution of this work was to define a discrete set of observation partitions on the multivariate continuous observation space via symbolic maximization techniques and derive the related probabilities using symbolic integration. An important avenue for future work is to determine whether similar techniques can be applied to the difficult case of continuous state, observation, *and* action DC-POMDPs.

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] J. Scott Penberthy and Daniel S. Weld. Temporal planning with continuous change. In *National Conference on Artificial Intelligence AAAI*, pages 1010–1015, 1994.

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

[7] 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.

[8] Pascal Poupart, Kee-Eung Kim, and Dongho Kim. Closing the gap: Improved bounds on optimal pomdp solutions. In *In Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, 2011.

[9] 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.

[10] 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.

[11] 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.

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