A Survey of Point-Based Algorithms for Solving POMDPs

Andrea Bajcsy

Abstract

A key problem in designing autonomous systems is taking input from the environment and producing actions that allow the system to reach a goal. In most real-world settings, actions have stochastic effects on the environment and the state of the environment is only partially observable. Partially Observable Markov Decision Processes (POMDPs) are an expressive and mathematically concrete framework that enable us to optimize decision problems with partial observability. However, POMDPs are hindered by computational intractability. Thus, good approximation techniques are needed in order to leverage the modeling power of POMDPs with realistic computation.

The goal of this survey is two-fold: (1) analyze the foundations of solving POMDPs with value iteration and point-based approximations and, (2) analyze a POMDP-based model of human internal state in the context of human-robot interaction.

1. Introduction

As autonomous systems begin to leave the assembly lines and actually interact with humans, there comes a need to model and predict humans actions and internal belief state in human-robot interaction settings. Partially Observable Markov Decision Processes (POMDPs) provide a solid mathematical framework for optimizing decision problems with partial observability. This is a desired characteristic in robotics applications where autonomous systems often operate under uncertain and dynamic environments. However, POMDPs are often computationally intractable, taking hours to compute an exact solution even for POMDPs with only a dozen states [3]. This leads to POMDPs being unusable for modeling realistic robotics problems. While computing exact solutions to POMDPs remains difficult, point-based

POMDP algorithms have provided fast approximate solutions for POMDPs even with hundreds of states [3].

1.1. Scope and Organization

In order to make POMDP models usable in interactive human-robot settings, a focus on good approximations of POMDPs is worth investigation. To lay the groundwork for understanding the design and usability of POMDP models for human-robot interaction, this preliminary literature survey provides a brief overview of point-based approximations for solving POMDPs and their use in modeling spoken language interaction with model uncertainty. First, we review the fundamentals of Markove Decision Processes (MDPs) versus POMDPs. Then, we summarize the fundamental leaps in point-based value iteration algorithms for approximating discrete and continuous POMDPs. Finally, we describe the use of value iteration approximations in the context of context of an adaptive human-robot interaction system where a spoken language system must adapt to a human user. We find that while approximate POMDP solving methods are becoming more and more efficient, newer trends in human-robot interaction emphasize that robots can actively probe the environment in order to gather more information about the partially observable states, rather than simply relying on passive observations.

1.2. Background: MDPs and POMDPs

Markov Decision Processes (MDPs) are at the base of solving complex Partially Observable Markov Decision Processes (POMDPs) and have been of interest to solving problems in robotic navigation, gesture recognition, factory control, and speech recognition systems [2], [5].

1.2.1. Markov Decision Processes

A MDP agent takes the state of the world as input and outputs actions that will in turn affect the state of the world. It can be described as a tuple $< S, A, T, R >$ where:

- S: set of the world states
- A: set of actions agent can take
- T: state-transition model defined by $P(s'|s, a)$ which denotes the probability distribution of the agent ending in state s', given that it was in state s and took action a

- R: reward function defined by $R(s, a)$ which denotes the expected reward the agent gains for taking action a in state s

It is important to note that in MDPs, the agent never has any uncertainty about it's state (although there may be uncertainty about the effects of the agent's actions on the world), and at time $t+1$, the state and reward depend only on the state and action at time t (Markov Property) [2].

Typically MDP algorithms focus on finite state and action cases, but POMDPs often consider a class of MDPs with infinite state spaces.

To act optimally, agents must maximize their long-term reward of each of their actions. Types of optimality are defined as:

- finite-horizon: agent acts to maximize expected sum of reward, R in the next N steps:
$$E[\sum_{t=0}^{N} R_t]$$
- discounted infinite-horizon: agent acts to maximize sum of rewards over infinite lifetime, but discounted by geometric factor $0 < \gamma < 1$:
$$E[\sum_{t=0}^{\infty} \gamma^t R_t]$$
  The larger the γ the more effect future rewards have on the current decision.

Policies determine the behavior of an agent at any state. Policies are evaluated based on the long-run value that the agent gets from utilizing it. This evaluation comes from the value function $V(s)$ that denotes the expected sum of reward from starting at state. Below, we have summarized the typical types of policies as well as their value functions:

- stationary: actions only depend on the current state and not on the time step: $\pi : S \to A$
  - stationary value function:
    $V(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s')V(s')$
- non-stationary: actions are defined as a sequence of state-to-action maps which are indexed by time: $\pi = \{\pi_t : s_t \to a_t,\ \forall\ t \in T\}$

  - non-stationary value function:
    $V_t(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V_{t-1}(s')$

While there are several exact methods by which we can find the optimal policy for MDPs including policy iteration and linear programming, we will focus on value iteration and point-based approximation variants in this survey. Value iteration has been explored not only in MDPs, but also for discrete and continuous-state POMDPs, and showed promise in scaling effectively to higher dimensional states for both discrete and continuous POMDPs [2], [4], [5].

1.2.2. Partially-Observable Markov Decision Processes

In the case of POMDPs, the agent can no longer determine it's current state with complete reliability. Thus, we cannot compute the optimal policy π and use it to act. Thus, to act truly optimally, a POMDP agent has to remember previous actions and observations to better distinguish states in it's world from previous states it has been in.

Similarly to the MDP agent, a POMDP agent can be described as a tuple $< S, A, T, R, O, \Omega >$ where the new O and Ω functions:

- T: state-transition model defined by:

$$T(b, a, b') = P(b'|a, b) = \sum_{o \in O} P(b'|a, b, o) P(o|a, b) \tag{1}$$

- R: reward function defined by:

$$R(b, a) = \sum_{s \in S} b(s) r(s, a) \tag{2}$$

- O: set of observations (providing incomplete information)
- Ω: observation model defined by $P(o'|s, a)$ which denotes the probability distribution of the agent observing o from state s after having taken action a

In order to remember previous actions and observations, a POMDP agent keeps an internal belief state, $b(s)$ = probability of world state s, encoding this previous experience. Belief states can be represented as probability distributions over states of the world and are a sufficient statistic for the agent's history [2]. In other words, if the agent computes it's current belief state then no new information about it's past states or actions would help it better understand its current state. The belief has to be updated to $b'(s)$ given an old belief $b(s)$, action a, and observation o by:

$$\begin{aligned} b'(s') = P(s'|o,a,b) &= \frac{P(o|s',a)\sum_{s\in S} P(s'|a,b,s)P(s|a,b)}{P(o|a,b)} \\ &= \frac{\Omega(s',a',o)\sum_{s\in S} T(s,a,s')b(s)}{P(o|a,b)} \end{aligned} \tag{3}$$

Finally, just like in MDPs, the value function for POMDPs allow us to compute an optimal policy. With value iteration we can construct the optimal non-stationary discounted value function across belief space. Let b be a belief and B be the set of all beliefs. Thus, the value function for a non-stationary, discrete-state POMDP can be represented as:

$$V_t(b) = \max_{a\in A}\{R(b,a) + \gamma \sum_{b'\in B} T(b,a,b')V_{t-1}(b')\} \tag{4}$$

2. Value Iteration

2.1. The Value Iteration Algorithm & Challenges

Although value iteration was proposed in the 1970s as a method for planning in POMDPs, many POMDP value iteration algorithms aren't scalable to real-world problems [4]. Two primary reasons underlie the difficulty in scaling value iteration algorithms:

1. Curse of dimensionality: If a problem has N states, then belief states are in $(N-1)$ dimensional space.
2. Curse of history: POMDP value iteration computes a set of action-observation histories, which grows exponentially with the planning horizon (think breadth-first search in belief space).

We can see that the higher the dimensionality of the belief space, the more action-observation histories it must compute and store. Point-based value iteration approximations seek to address the curse of history by selecting a small but representative set of belief points and applies the value iteration updates to these points. This update becomes quadratic rather than exponential which allows for greater scalability.

The general value iteration algorithm for POMDPs can be seen in Algorithm 1. Line 5 is called the Bellman update (or Bellman back-up) and $V_t(b)$ represents the expected sum of rewards starting with belief b and acting optimally for H steps.

Algorithm 1 Exact Value Iteration Algorithm

```
1: $V_0(s) = 0, \forall s$
2: for $t = 1, 2, ..., H$ do
3:     for all $s \in S$ do
4:         $V_{t+1}(b) = \max_{a \in A}\{R(b,a) + \gamma \sum_{b' \in B} T(b,a,b')V_t(b')\}$
5:     end for
6: end for
```

After H iterations, the solution will consist of a set of α-vectors, each of which represent the optimal policy assuming all the following $(H-1)$ steps are optimal too. In the exact computation of the value function, it is always piecewise linear and convex in the belief [2], [4]. See Figure 1 for simple visualization of POMDP value functions. Note here that many of the computed α-vectors have empty maximizing regions and are therefore useless in computing the optimal value function! While the pruning of such α-vectors has been studied, it is generall expensive to identify and prune them from the set [6].

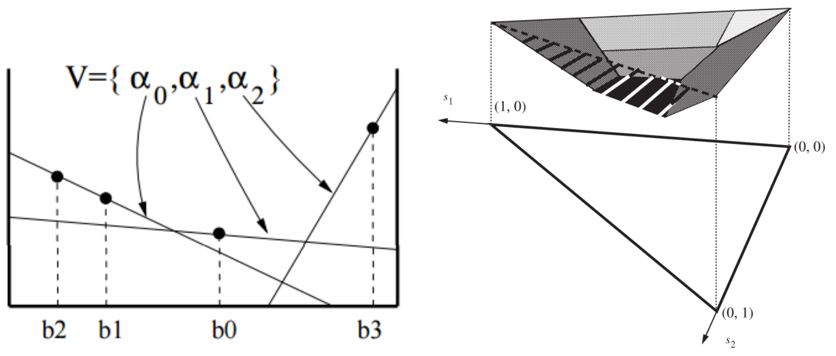

Figure 1: Illustration of 2D and 3D POMDP value functions. Each α-vector forms a hyperplane with dimension equal to the number of states. The optimal value function takes the drawn "bowl" shape. Images from [4] (left), and [2] (right).

2.2. Point-Based Value Iteration for POMDPs

In [4], the authors introduce the Point-Based Value Iteration (PBVI) algorithm for POMDP planning with a finite set of belief points. By selecting a small, representative set of belief points, this algorithm approximates the exact value iteration solution. Additionally, by using stochastic trajectories to choose belief points, PBVI is able to solve larger problems that exact value iteration methods cannot.

The intuition behind the PBVI algorithm is that an agent should only spend time computing solutions to parts of the belief space that can actually be encountered by interacting with the environment. Thus, the PBVI algorithm starts with an initial set of belief points, $B = \{b_0, b_1, ..., b_q\}$. The algorithm initializes an α-vector for each belief point and uses a modified version of the Bellman update to update the value of the α-vector. Note, since PBVI preserves the full α-vector, it ensures that the properties of piecewise linearity and convexity are preserved. The algorithm then grows the set of belief points, finds a solution, and repeats. Two questions remain: (1) how to select belief points, and (2) how to perform the efficient updates.

2.2.1. Selecting belief points

Initially, the set B contains the initial belief b_0. To select new belief points for any given b, PBVI first stochastically simulates a step forward for each available action $a \in A$ and produces $B' = \{b_{a_0}, b_{a_1}, ...\}$. To simulate an action a, the algorithm randomly samples a state s from the belief distribution b, then samples observation according to $\Omega(s, a, o)$, and computes b_a with the Bayesian update in Equation 4.

To determine which sampled beliefs to keep, for each $b' \in B'$, the algorithm measures the L_1 distance from each $b \in B$ and discards b' from the set if $b' \in B$. The resulting belief points that are kept are farthest away from any points that were already in original set B. At each expansion step, the new belief set at most doubles in size.

2.2.2. Efficient point-based value updates

The traditional Bellman update in Algorithm 1 is modified such that only one α-vector is kept for each belief point. Due to this, the PBVI algorithm only considers beliefs that maximize the value function for at least one of the examples, visually demonstrated in Figure 2.

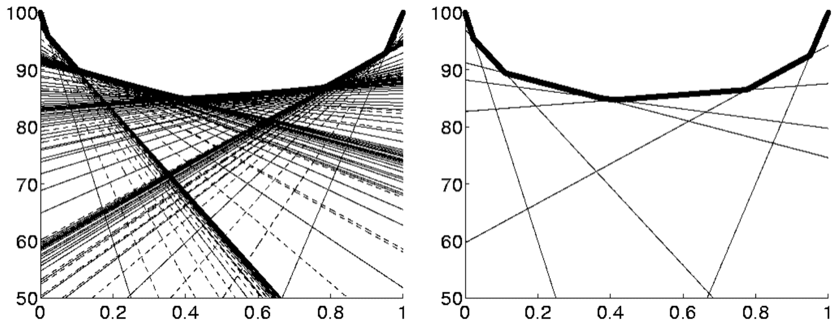

Figure 2: Comparison of exact value function computation (left) in 2D vs. point-based value iteration computation (right). Images from [7].

3. Randomized Point-Based Value Iteration For POMDPs

In [6], the authors extended the original idea of point-based value iteration approximation of POMDPs from [4] to enable only updating a randomly selected subset of points in the belief set. Additionally, they show extensions from discrete action spaces to continuous action spaces as well as experimental results in large scale POMDP problems.

The PERSEUS algorithm presented in [6] follows a similar paradigm as the orginal PBVI paper but only a random subset of belief points get updated. The claim is that a single update can ultimately improve many points in the belief set.

4. Application: Spoken Language Human-Robot Interaction System Modeled as POMDP

[1]

5. Discussion

6. Conclusion

References

[1] Doshi, F., and Roy, N. Spoken language interaction with model uncertainty: an adaptive human–robot interaction system. *Connection Science 20*, 4 (2008), 299–318.

[2] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*, 1 (1998), 99–134.

[3] Kurniawati, H., Hsu, D., and Lee, W. S. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems* (2008), vol. 2008, Zurich, Switzerland.

[4] Pineau, J., Gordon, G., Thrun, S., et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI* (2003), vol. 3, pp. 1025–1032.

[5] Porta, J. M., Vlassis, N., Spaan, M. T., and Poupart, P. Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research 7*, Nov (2006), 2329–2367.

[6] Spaan, M. T., and Vlassis, N. Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research 24* (2005), 195–220.

[7] Weld, D. Partially observable mdps (pomdps), 2012.