Delayed Observation Planning in Partially Observable Domains

(Extended Abstract)

Pradeep Varakantham[†], Janusz Marecki[‡]
[†]School of Information Systems, Singapore Management University, Singapore
[‡]IBM Watson Research Lab, New York, NY
[†]pradeepv@smu.edu.sg [‡]marecki@us.ibm.com

ABSTRACT

Traditional models for planning under uncertainty such as Markov Decision Processes (MDPs) or Partially Observable MDPs (POMDPs) assume that the observations about the results of agent actions are instantly available to the agent. In so doing, they are no longer applicable to domains where observations are received with delays caused by temporary unavailability of information (e.g. delayed response of the market to a new product). To that end, we make the following key contributions towards solving Delayed observation POMDPs (D-POMDPs): (i) We first provide an parameterized approximate algorithm for solving D-POMDPs efficiently, with desired accuracy; and (ii) We then propose a policy execution technique that adjusts the policy at runtime to account for the actual realization of observations. We then show the performance of our techniques on POMDP benchmark problems with delayed observations where explicit modeling of delayed observations leads to solutions of superior quality.

Categories and Subject Descriptors

G [**3**]: Markov Processes

General Terms

Algorithms

Keywords

Partially Observable Markov Decision Process, Delayed Observations

1. INTRODUCTION

Recent years have seen a rise of interest in autonomous agents deployed in domains ranging from automated trading, traffic control, disaster rescue and space exploration. Simultaneously, research in devising control mechanisms for these agents has progressed significantly. Partially Observable Markov Decision Processes (POMDPs) have received considerable attention, due to their ability to capture the uncertainty of the outcomes of agent actions and the uncertainty in the agent observations of the environment. Research in POMDPs has allowed the POMDP solvers [5] to scale to domains with thousands of states, actions and observations while providing guarantees on solution quality.

Unfortunately, the problem of decision making with delayed observations has received scant attention in POMDP research. Delayed observation reasoning is particularly relevant in providing real time decisions based on traffic congestion/incident information [1] or in making decisions on new products before receiving the market response to a new product. There are always delays in receiving such information due to data fusion, computation, transmission and physical limitations of the underlying process. Existing research [1, 3] has provided (a) models to represent observation delay in the context of Markov Decision Problems; (b) Theoretical properties of the sufficient statistic and value function; and (c) optimal approaches for solving MDPs with fixed observation delays;. While those models and algorithms are extended to POMDPs, the optimal nature of the algorithms and other restrictions (such as fixed observation delays) decreases their scalability and applicability.

In this paper we remedy the shortcomings of the previous work for handling delayed observations, in three key contributions: (i) We provide a parameterized approximate algorithm for solving D-POMDPs with a desired accuracy; (ii) We propose a policy execution technique that adjusts the agent policy corresponding to delayed observations at runtime for improved performance; and (iii) Finally, we provide error bounds, theoretical properties and complexity results for the proposed approaches. In the experimental results, we illustrate that our planning and execution algorithms lead to improved performance in domains with observation delays.

2. MODEL: D-POMDP

We now introduce D-POMDP model to allow for rich modeling of delayed observations, extending the models proposed in [1, 3]. A D-POMDP is a tuple $\langle S, A, \Omega, P, R, O, \mathcal{X} \rangle$ whose only difference from a POMDP is $\mathcal{X}$—a set of random variables $\mathcal{X}_{s,a}(k)$ that specify the probability that an observation is delayed by k decision epochs, when action a is executed in state s. *An example of $\mathcal{X}_{s,a}$ would be the discrete distribution (0.5,0.3,0.2), where 0.5 is the probability of no delay, 0.3 is the probability of one time step delay and 0.2 is the probability of two time step delay in receiving the observation in state s on executing action a.* D-POMDPs extend POMDPs by modeling the observations that are de-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

layed and by allowing for actions to be executed prior to receiving these delayed observations. In essence, if the agent receives an observation immediately after executing an action, D-POMDPs behave exactly as POMDPs. However, if an observation does not reach the agent immediately, D-POMDPs behave differently from POMDPs. Rather than having to wait for an observation to arrive, a D-POMDP agent can resume the execution of its policy *prior to* receiving the observation. In short, a D-POMDP agent must balance the trade off of acting prematurely (without the information provided by the observations that have not yet arrived) versus executing stop gap (waiting) actions.

Our introduction of D-POMDPs is accompanied in the next section by a D-POMDP example for a classical "Tiger Domain" [4] wherein $S = \{s_{TigerLeft}, s_{TigerRight}\}$, $A = \{a_{OpenLeft}, a_{OpenRight}, a_{Listen}\}$, $O = \{o_{TigerLeft}, o_{TigerRight}\}$ and the observations $o_{TigerLeft}$, $o_{TigerRight}$ resulting from the execution of action a_{Listen} arrive with a delay sampled from a discrete probability distribution $\mathcal{X}$.

3. SOLVING D-POMDPS

In this paper, we are interested in providing quality bounded and efficient solutions for D-POMDPs. Our approach to solving a D-POMDP consists of two steps: (a) converting the D-POMDP to an approximately equivalent POMDP; and (b) employing an existing POMDP solver to solve the obtained POMDP. The key step is (a) and we provide a parameterized approach for making the conversion from D-POMDP to its approximately equivalent POMDP. The level of approximation is governed by an input parameter, D, which represents the number of delay steps considered in the planning process[1] . The extended POMDP obtained from the D-POMDP is defined as the tuple $\langle \overline{S}, A, \overline{\Omega}, \overline{P}, \overline{R}, \overline{O} \rangle$ where $\overline{S}$ is the set of extended states and $\overline{\Omega}$ is a set of extended observations that the agent receives upon executing its actions in extended states. $\overline{P}, \overline{R}, \overline{O}$ are the extended transition, reward and observations functions.

3.1 Online Policy Modification

The second key contribution of this paper is a technique for modifying the policy of a converted POMDP (from previous section) during execution. We assume here that the employed POMDP solver returns value vectors (along with dominating actions) across the belief space. Typically, the policy execution in a POMDP is initiated by executing the action at the root of the policy tree, selecting and executing the next action based on the received observation and so on. This type of policy execution suffices in normal POMDPs, however, in extending POMDPs corresponding to D-POMDPs, the policy execution can be improved. The key intuition here is that during policy execution the beliefs that an agent has can be outdated (due to not updating the belief once observations are delayed). Hence, the idea is to keep the belief state as updated as possible in an efficient manner, i.e. updating the beliefs as and when the delayed observations are received.

4. EXPERIMENTS

[1]At execution time, we can receive observations at delays greater than D

In this paper, we have experimented with different types of problems to evaluate the performance of our planning and execution algorithms. Since state-of-the-art POMDP solvers [5] are already capable of solving problems involving hundreds of thousands of states and we hope to implement our techniques on top of those approaches. However, in this paper, we will be focusing mainly on understanding how much our planning and execution approaches can improve with respect to quality as the delay distribution increases in complexity.

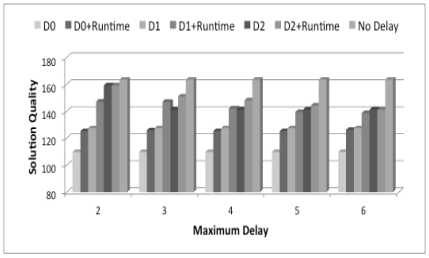

Figure 1: Comparison of solution quality

We experimented with benchmark problems in POMDP literature with different delay distributions. The main problems that we experimented with include: "tiger", "1d maze", "network", "4x4.95.POMDP" and "paint95", taken from Anthony Cassandra's POMDP page. Observation delay is defined by the set of discrete distributions $\mathcal{X}$ in the D-POMDP model. Our planning algorithms are represented as D0, D1, D2 etc., where the number corresponds to the D employed in the conversion of D-POMDP to POMDP. We compare the solution quality obtained by D0, D1 and D2 with and without the online policy modification component (Runtime) against the solution quality obtained if there was no delay in receiving observation. To solve the converted POMDP problems we employ the Point Based Value Iteration solver [2], but any of the existing solvers can be employed. Figure 1 shows the performance of our algorithms as the maximum possible observation delay, Δ, is increased. The problems are categorized according to the value of $\{\mathcal{X}(0) + \mathcal{X}(1)\}$.

5. REFERENCES

[1] J.L. Bander and C.C. White. Markov decision processes with noise corrupted and delayed state observations. *Journal of OR Society*, 50:660–668, 1999.

[2] G. Gordon J. Pineau and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference in Artificial Intelligence*, 2003.

[3] K.V. Katsikopoulos and S. E. Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE transactions on automatic control*, 48:568–574, 2003.

[4] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[5] T. Smith and R. G. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *UAI*, 2005.