

Using Rewards for Belief State Updates in Partially Observable Markov Decision Processes

Masoumeh T. Izadi and Doina Precup

McGill University, School of Computer Science,
3480 University St., Montreal, QC, Canada, H3A2A7

Abstract. Partially Observable Markov Decision Processes (POMDP) provide a standard framework for sequential decision making in stochastic environments. In this setting, an agent takes actions and receives observations and rewards from the environment. Many POMDP solution methods are based on computing a *belief state*, which is a probability distribution over possible states in which the agent could be. The action choice of the agent is then based on the belief state. The belief state is computed based on a model of the environment, and the history of actions and observations seen by the agent. However, reward information is not taken into account in updating the belief state. In this paper, we argue that rewards can carry useful information that can help disambiguate the hidden state. We present a method for updating the belief state which takes rewards into account. We present experiments with exact and approximate planning methods on several standard POMDP domains, using this belief update method, and show that it can provide advantages, both in terms of speed and in terms of the quality of the solution obtained.

1 Introduction

Sequential decision making and control problems in dynamic environments with incomplete and uncertain information have been the focus of many researchers in different disciplines. Designing agents that can act under uncertainty is mostly done by modelling the environment as a Partially Observable Markov Decision Process (POMDP) [2]. In POMDPs, an agent interacts with a stochastic environment at discrete time steps. The agent takes actions, and as a result, receives observations and rewards. The agent then has to find a way of choosing actions, or policy, which maximizes the total reward received over time. Most POMDP planning methods try to construct a Markovian state signal using a model of the environment and the history of actions and observations experienced by the agent. This signal is called a belief state. Planning methods then use reward information in order to associate an (optimal) action to each belief state.

The fact that rewards are only used in the computation of the optimal policy, but not in updating the belief state, is due to the fact that POMDPs have roots in Markov Decision Processes (MDPs). In an MDP, the agent is provided with Markovian state information directly, and uses the rewards in order to obtain an optimal policy. The same scenario has been carried out to POMDP planning methods as well, except now the Markovian belief state has to be recovered from the history. However, rewards are still used only in policy computation, and not in updating the belief state. Intuitively, it seems

that rewards can carry useful information, which can help the agent guess the hidden state more precisely. Indeed, in related work, James et al. [7] used rewards to update information about the state of an agent, in the context of predictive state representations (PSRs). They noted that for some of the domains used in their experiments, using the reward information seemed to help in finding better policies. Our goal in this paper is to investigate whether rewards can be used to produce a similar effect using traditional POMDP planning methods. Intuitively, in some tasks, rewards can provide additional information, not captured by observations. In this case, we expect that using rewards could result in a better estimate of the belief state, and perhaps, in belief states that can identify more precisely the hidden state of the system. If the hidden state were known with better precision, the action choices of the agent could be better as well.

In this paper we describe how rewards can be used for updating belief states. We evaluate empirically the merit of this method, compared with usual belief updates, on several POMDP benchmarks, using both exact and approximate planning, and find that using rewards can be beneficial, if they contained additional information that is not captured in the observations. We find that using rewards can help decrease the entropy of the belief states.

2 Partially Observable Markov Decision Processes

Formally, a POMDP is defined by the following components: a finite set of hidden states S ; a finite set of actions A ; a finite set of observations Z ; a transition function $T : S \times A \times S \rightarrow [0, 1]$, such that $T(s, a, s')$ is the probability that the agent will end up in state s' after taking action a in state s ; an observation function $O : A \times S \times Z \rightarrow [0, 1]$, such that $O(a, s', z)$ gives the probability that the agent receives observation z after taking action a and getting to state s' ; an initial belief state b_0 , which is a probability distribution over the set of hidden states S ; and a reward function $R : S \times A \times S \rightarrow \mathfrak{R}$, such that $R(s, a, s')$ is the immediate reward received when the agent takes action a in hidden state s and ends up in state s' . Additionally, there can be a discount factor, $\gamma \in (0, 1)$, which is used to weigh less rewards received farther into the future.

The goal of planning in a POMDP environment is to find a way of choosing actions, or policy, which maximizes the expected sum of future rewards $E[\sum_{t=0}^T \gamma^t r_{t+1}]$ where T is the number of time steps left to go in a finite horizon problem, or ∞ in an infinite horizon problem. The agent in a POMDP does not have knowledge of the hidden states, it only perceives the world through noisy observations as defined by the observation function O . Hence, the agent must keep a complete history of its actions and observations, or a sufficient statistic of this history, in order to act optimally. The sufficient statistic in a POMDP is the belief state b , which is a vector of length $|S|$ specifying a probability distribution over hidden states. The elements of this vector, $b(i)$, specify the conditional probability of the agent being in state s_i , given the initial belief b_0 and the history (sequence of actions and observations) experienced so far. After taking action a and receiving observation z , the agent updates its belief state using Bayes' Rule:

$$b'(s') = P(s'|b, a, z) = \frac{O(a, s', z) \sum_{s \in S} b(s) T(s, a, s')}{P(z|a, b)} \quad (1)$$

The denominator is a normalizing constant and is given by the sum of the numerator over all values of $s' \in S$:

$$P(z|a,b) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s,a,s') O(a,s',z)$$

We can transform a POMDP into a “belief state MDP” [2]. Under this transformation, the belief state b becomes the (continuous) state of the MDP. The actions of the belief MDP are the same as in the original POMDP, but the transition and reward functions are transformed appropriately, yielding the following form of Bellman optimality equation for computing the optimal value function, V^* :

$$V^*(b) = \max_{a \in A} \sum_{z \in Z} P(z|a,b) \left[\sum_{s \in S} b(s) \left(\sum_{s'} b'(s') R(s,a,s') \right) + \gamma V^*(b') \right]$$

where b' is the unique belief state computed based on b , a and z , as in equation (1). As in MDPs, the optimal policy that the agent is trying to learn is greedy with respect to this optimal value function. The problem here is that there are infinite number of belief states b , so solving this equation exactly is very difficult.

Exact solution methods for POMDPs take advantage of the fact that value functions for belief MDPs are piecewise-linear and convex functions, and thus can be represented by a finite number of hyperplanes in the space of beliefs. Value iteration updates can be performed directly on these hyperplanes. Unfortunately, exact value iteration is intractable for most POMDP problems with more than a few states, because the size of the set of hyperplanes defining the value function can grow exponentially with each step. Approximate solution methods usually rely on maintaining hyperplanes only for a subset of the belief simplex. Different methods use different heuristics in order to define which belief points are of interest (e.g. [1],[5], [9], [11]).

3 Using Rewards in the Computation of Belief States

Reward signals may provide useful information about the true state of a POMDP system. Although many of the POMDP benchmarks used to evaluate POMDP planning algorithms (e.g., from Tony Cassandra’s repository [3]) have been designed in such a way that rewards do not carry any additional information that is not contained in the observations, extra information could potentially be present. In this section, we describe a straightforward way of using rewards in the process of updating belief states.

As described in the previous section, rewards in POMDPs are given as a function $R(s,a,s')$, which depends on the current state, the next state and the action taken. Hence, we can treat rewards as random variables, with a conditional probability distribution $P(r|s,a,s')$, where $P(r|s,a,s') = 1$ if and only if $r = R(s,a,s')$, and 0 otherwise. Note that if we had additional information about the distribution of immediate rewards, instead of just knowing the expected values, this could be naturally incorporated in this framework.

If there is only a discrete, finite set of reward values possible in the MDP, $\{r_1, r_2, \dots, r_k\}$, where each r_i represents the immediate reward for taking some action a from some hidden state s , this probability distribution can be easily specified using a

table. We note that in most POMDP examples, the number of possible immediate rewards satisfies this assumption, and is often very small. However, if this assumption is not satisfied, e.g. if reward are continuous, a conditional probability distribution over rewards can still be specified in some parametric form, based on the given model of the POMDP.

We note that rewards and observations are conditionally independent given the current state s , the next state s' and the action a . From now on we will treat rewards in the same way as observations in predictions about the future. The definition of a history will be extended to include the rewards: $h = a_1 z_1 r_1 \dots a_n z_n r_n$.

We define belief state updates based on rewards and observations, by analogy with equation (1), as follows:

$$\begin{aligned} b'(s') &= P(s'|r, z, a, b) = \frac{P(b, a, s', r, z)}{P(b, a, r, z)} \\ &= \frac{\sum_{s \in \mathcal{S}} b(s) T(s, a, s') O(a, s', z) P(r|s, a, s')}{\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} b(s) T(s, a, s') O(a, s', z) P(r|s, a, s')} \end{aligned}$$

Value functions are computed in this case analogously to the case of regular beliefs. We will call this model Reward-based POMDP (RPOMDP).

Rewards have been used by Poupart and Boutilier [10] in designing a compression method for POMDPs. The value-directed POMDP compression algorithm of Poupart and Boutilier is close to RPOMDP model. This algorithm computes a low dimensional representation of a POMDP directly from the model parameters R, T, O by finding Krylov subspaces for the reward function under belief propagation. The Krylov subspace for a vector and a matrix is the smallest subspace that contains the vector and is closed under multiplication by the matrix. The authors use the smallest subspace that contains the immediate reward vector and is closed under a set of linear functions defined by the dynamics of the POMDP model. James et al's variation of PSRs in [7] also represents reward as a part of observation. It is interesting to note that this definition of PSRs makes it very similar to value-directed compression [10] for POMDPs.

4 Empirical Evaluations

In this section we focus on studying the effect of using rewards in belief state updates. We selected five standard domains from the POMDP repository [3] and from [10]. Table 1 lists these problems with their characteristics. We chose three domains in which the rewards provide additional information compared to observations. These domains are: Network, line4-2goals and coffee. The other two domains are ones in which rewards possess the same information as the observations (4x4 grid world) or rewards have more information than observations only for special actions (shuttle).

We performed a set of experiments to test the effect of rewards in reducing the uncertainty about the hidden states on selected domains. The entropy of the agent's beliefs have been measured for both the POMDP and RPOMDP model on 100 time steps running the same random policy. Figure 1 shows the result of this experiment in the five domains. The graphs are averages taken over 5 independent runs. For the network,

Table 1. Domains used in the experiments

Domain	$ S $	$ \mathcal{A} $	$ O $	$ \mathcal{R} $
line4-2goals	4	2	1	2
network	7	4	2	6
4x4 grid	16	4	2	2
shuttle	8	3	5	3
coffee	32	2	3	12

Table 2. Performance comparison of exact solutions for POMDP original model and RPOMDP model

Domain	infinite horizon	infinite horizon	finite horizon	finite horizon
	POMDP	RPOMDP	POMDP	RPOMDP
line4-2goals				
time	0.01	0.01	0.00	0.00
α -vectors	2	2	2	2
reward	0.466	0.466	0.465	0.465
network				
time	4539.15	5783.02	2.89	3.24
α -vectors	487	549	197	216
reward	293.185	340.15	121.27	173.94
coffee				
time	-	6.35	103	0.23
α -vectors	N/A	5	263	5
reward	N/A	0.00	0.00	0.00
4x4 grid				
time	5578.5	6459.3	231	316
α -vectors	243	243	245	248
reward	1.209	1.209	1.073	1.073
shuttle				
time	-	-	1270	81.4
α -vectors	N/A	N/A	2011	1152
reward	N/A	N/A	11.974	11.237

line4-2goals and coffee domains, the uncertainty of RPOMDP beliefs decreases considerably and it stays always lower than the entropy of POMDP beliefs. Shuttle and 4x4 grid do not show noticeable difference for POMDP and RPOMDP running a random policy. This is expected since the rewards carry little more or no more information than the observations for some of the actions.

In a second set of experiments, we used exact and approximate POMDP solution techniques to evaluate the performance of the RPOMDP model in terms of time to reach an optimal solution, the reached optimal value functions, and the complexity of the optimal value function with respect to the number of alpha vectors used to represent it.

The fastest exact solution method for POMDPs is the *Witness* algorithm [2]. We ran this algorithm on all of the domains, but an optimal solution could not be found in a rea-

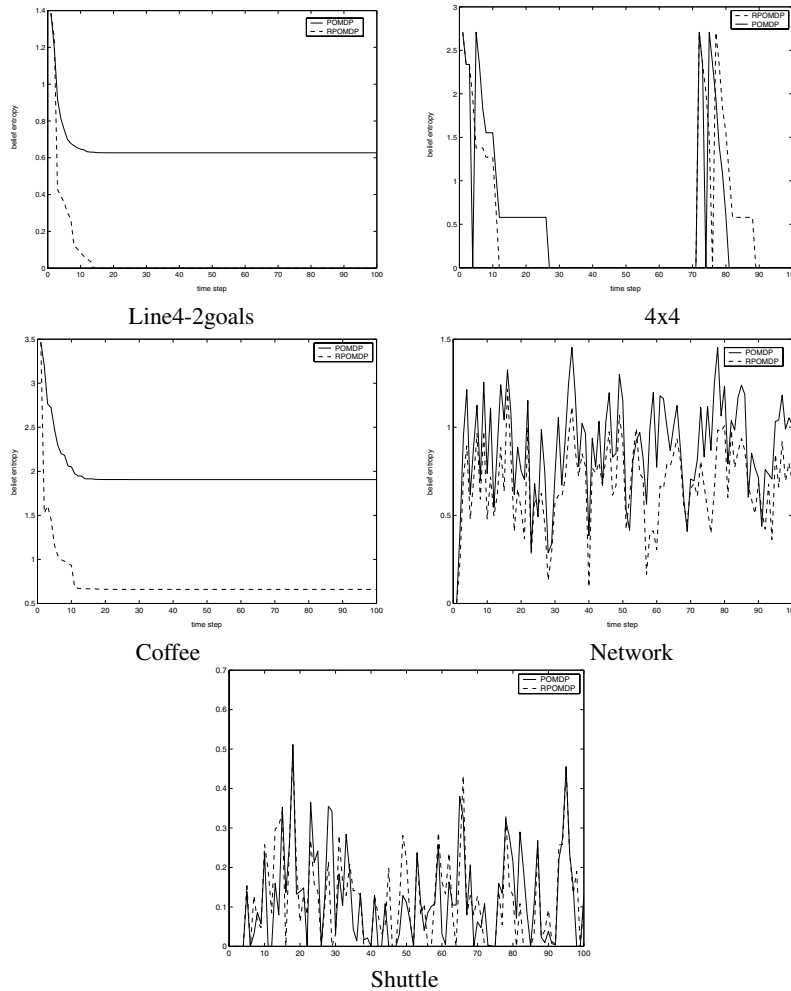


Fig. 1. Comparison of the uncertainty in the agent’s state of the world, using standard POMDP belief update vs. RPOMDP belief update, for 5 different tasks

sonable time for the shuttle and coffee domains. Table 2 shows a comparison between POMDP and RPOMDP for updating the belief states, when the Witness algorithm is used to find an optimal policy for a finite horizon 10 as well as for an infinite horizon. The reward reported in this table corresponds to the value of an initial belief state drawn uniformly. The dashes in the table show the cases where the algorithm has not been able to solve the problem in 5 hours. The results for the coffee domain are quite interesting. The Witness algorithm cannot perform more than 14 iterations in the time allowed with the usual belief state update, but can solve the problem very quickly when the belief update includes rewards as well. It turns out that the true beliefs for this problem occur on a very low dimensional manifold as it has been stated in [10] and it seems that RPOMDP can take advantage of this problem structure.

One of the most promising approaches for finding approximate POMDP value functions are point-based methods. In this case, instead of optimizing the value function over the entire belief space, only specific beliefs are considered. In our experiments we used the PBVI algorithm [9] together with regular and reward-based belief state updates. In PBVI, a finite set of reachable belief points is selected heuristically, and values are computed only for these points. The algorithm has an anytime flavor, adding more points over time to increase the accuracy of the value function representation. The results of the evaluation are shown in Table 3. We have ran the PBVI algorithm for 10 iterations for each domain and the results are averages over 5 independent runs. The results are obtained when starting with a specified initial belief. In all cases, we performed 10 belief set expansion of PBVI to obtain a value function. Then, we ran 250 trials of the standard PBVI algorithm. Each trial runs for 250 steps. We averaged the results obtained over these trials, and over 5 independent runs. The results in Table 3 confirm that for the problems in which rewards possess information about states more than observations, a significant improvement can be achieved with respect to time, quality of the solution obtained, or both. For the cases in which rewards do not help reduce the belief state entropy, there is no gain. Although RPOMDP in these cases might not sacrifice the solution quality, it can increase the computation time.

Table 3. Performance comparison of approximate solutions for POMDP original model and RPOMDP model

Domain	POMDP	RPOMDP
line4-2goals		
time	0.00	0.00
α -vectors	2	2
beliefs	232	157
discounted reward	1.24	0.48
network		
time	2	62
α -vectors	23	208
beliefs	714	500
discounted reward	240.63	352.9
coffee		
time	4	6
α -vectors	24	6
beliefs	323	169
discounted reward	-1.92	-0.97
shuttle		
time	0.8	0.01
α -vectors	17	18
beliefs	122	125
discounted reward	329.56	32.96
4x4 grid		
time	2.4	8.2
α -vectors	24	24
beliefs	460	468
discounted reward	3.73	3.75

5 Conclusions and Future Work

In this paper we studied the probabilistic reformulation of the POMDP model, focusing on the assumption that rewards carry information about the states of the world independent of observations. Following this assumption we represent and update belief states taking into account the reward signal as part of the feedback that the agent receives from the environment in order to reduce the uncertainty about the state of the world. We presented the results of an empirical study confirming that the RPOMDP model is very useful in reducing the entropy of beliefs for some domains. In this case, better solutions can be obtained as well, and computation time is typically reduced.

This research can be extended in several directions. This model can be used to study from a different perspective the space of reachable beliefs. Belief state entropy can be used in order to guide more intelligent exploration strategies. In practical problems, often the evolution of beliefs following a trajectory is embedded in a low dimensional space. In the case of RPOMDP, linear dimensionality reduction involves no information loss. We plan to study further how this way of updating beliefs affect non-linear compression algorithms.

References

1. Bonet, B. (2002) An epsilon-optimal grid-based algorithm for partially observable Markov decision processes. *Proceedings of ICML*, 51-58.
2. Cassandra, A., T., Littman, M., and Kaelbling L., P. (1997) A simple, fast, exact methods for partially observable Markov decision processes. *Proceedings of UAI*.
3. Cassandra, A. T. Tony's POMDP page (1999).
<http://www.cs.brown.edu/research/ai/pomdp/code/index.html>
4. Givan, R., Dean, T., and Greig, M. (2003) Equivalence notions and model minimization in Markov Decision Processes. *Journal of Artificial Intelligence*, 147:163-223(61).
5. Hauskrecht, M. (2000) Value-function approximation for partially observable Markov decision process. *Journal of Artificial Intelligence Research* 13:33-94.
6. Izadi, M. T., Rajwade, A., and Precup, D. (2005) Using core beliefs for point-based value iteration. *Proceedings of IJCAI*.
7. James, M., R., Singh, S., and Littman, M., L. (2004) Planning with Predictive State Representations. In *Proceedings of ICML*, 304-311.
8. Littman, M. L., Sutton, R., and Singh, S. (2002) Predictive representations of state. In *Proceedings of NIPS*, 1555-1561.
9. Pineau, J., Gordon, G., and Thrun, S. (2003) Point-based value iteration: An anytime algorithms for POMDPs. *Proceedings of IJCAI*, 1025-1032.
10. Poupart, P., and Boutilier, C. (2002) Value-directed Compression of POMDPs. *Proceedings of NIPS*, 1547-1554.
11. Smith, T., and Simmons, R. (2004) Heuristic search value iteration for POMDPs. *Proceedings of UAI*.