CSCI 2951P Final Report: iPOMDP

Alexis Cook, Gabe Hope, and Geng Ji

December 17, 2015

Introduction

A partially observable Markov decision process (POMDP) is a generalization of a Markov decision process, wherein an agent cannot directly observe states. Formally, a POMDP is a tuple $\{S, A, O, R, T, \Omega, \mathcal{R}, \gamma\}$. S, A, O, and R are sets of world states, actions, observations, and rewards. $\mathcal{T} \in [0,1]^{|S|\times|A|\times|S|}$, $\Omega \in [0,1]^{|S|\times|A|\times|O|}$, and $\mathcal{R} \in [0,1]^{|S|\times|A|\times|R|}$ are tensors defining conditional transition probabilities between states, conditional observation distributions, and conditional reward distributions. $\gamma \in [0,1]$ is the discount factor.

At time point t, the agent is in some unknown state s_t. The agent then takes some action a_t and transitions to a new unknown state s_{t+1}, which is a draw from a categorical distribution with probabilities specified in $\mathcal{T}(s_t, a_t, \cdot)$. At the same time, the agent receives an immediate reward r_t, which is a draw from a categorical distribution with probabilities specified in $\mathcal{R}(s_t, a_t, \cdot)$. Immediately after arriving in state s_{t+1}, the agent receives an observation o_{t+1}, which is a draw from a categorical distribution with probabilities specified in $\Omega(s_{t+1}, a_t, \cdot)$. The iPOMDP presumes that the agent does not know the total number of world states ($|S|$) a priori; instead, the state space S will be learned through experience, with no restrictions on its cardinality. The set of actions A, observations O, and rewards R are known to the agent. Along with S, the agent learns $\mathcal{T}$, Ω, and $\mathcal{R}$ through its interactions with the environment.

At every time step t, given a history $h_t = \{a_0, o_0, r_0, \ldots, a_t, o_t, r_t\}$, the agent infers an estimate of the underlying POMDP to choose the next action a_{t+1} that will maximize the expected discounted future reward $\mathbb{E}\sum_{s=t+1}^{\infty} \gamma^{s-(t+1)} r_s$. Updated estimates for S, $\mathcal{T}$, Ω, and $\mathcal{R}$ are acquired at every time step.

We approach inference through the use of the generative model introduced by Doshi-Velez [1]:

$$\begin{aligned} \overline{\mathcal{T}} &\sim \text{GEM}(\lambda) \\ \mathcal{T}(s,a,\cdot) &\sim \text{DP}(\alpha, \overline{\mathcal{T}}) \ \forall s \in S, a \in A \\ \Omega(s,a,\cdot) &\sim H \ \forall s \in S, a \in A \\ \mathcal{R}(s,a,\cdot) &\sim H_R \ \forall s \in S, a \in A \end{aligned} \tag{1}$$

This model (which we have sketched ourselves from the description in [1]) is depicted in Figure 1. Here, the set of conditional transition probabilities $\{\mathcal{T}(s,a,\cdot) : s \in S, a \in A\}$ is composed of i.i.d draws from a Dirichlet process with concentration parameter α and base distribution $\overline{\mathcal{T}}$. The mean transition distribution function $\overline{\mathcal{T}}$ is a draw from a Dirichlet process with concentration parameter λ. Both $\lambda \in \mathbb{R}_{>0}$ and $\alpha \in \mathbb{R}_{>0}$ assume predetermined values. H and H_R are Dirichlet priors for the observations and reward distribuitions that are specified prior to initiating the inference loop. The set of conditional observation distributions $\{\Omega(s,a,\cdot) : s \in S, a \in A\}$ and conditional reward distributions $\{\mathcal{R}(s,a,\cdot) : s \in S, a \in A\}$ are draws from H and H_R, respectively. Although we choose the paper in [1] for replication, we borrow important ideas from [4].

Implementation

The pseudocode in [1] was minimal (spanning only roughly five or ten lines in length) and omitted discussion of many algorithmic details. To make sense of the paper, we met extensively to work out points of confusion and produced detailed pseudocode, which appears in the *Appendix*. It is still not entirely clear to us exactly how our algorithm compares to the algorithm in [1], but we figure that our detailed pseudocode would be useful for potentially discussing our ideas with the original author at a later date. The algorithm can be coarsely described as follows, where T is the total number of time steps for which the agent interacts with the environment:

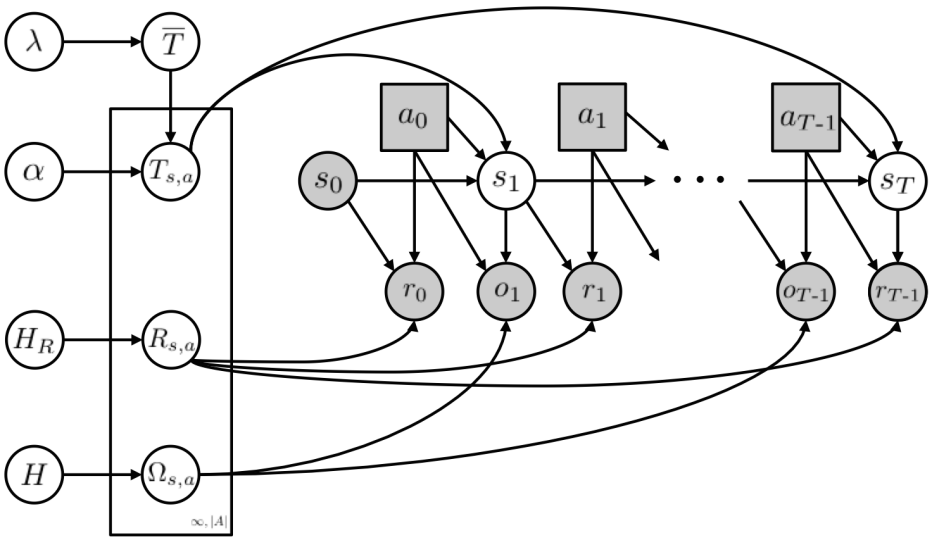


Figure 1: Generative Model

Algorithm *Main Loop*

1. **for** $t \leftarrow 0$ **to** T
2. **do** Update posterior estimate on iPOMDP parameters $\{S, \mathcal{T}, \Omega, \mathcal{R}\}$ using h_t
3. **do** Using iPOMDP parameters, select and take action a_{t+1} that will maximize expected discounted future reward; receive o_{t+1} and r_{t+1}

The authors use beam sampling to infer the iPOMDP parameters. The main idea behind the beam sampler is that at some timestep t, we view $h_t = \{a_0, o_0, r_0, \ldots, a_t, o_t, r_t\}$ as a set of observed random variables, and resample a set of latent variables $\{S, \mathcal{T}, \Omega, \mathcal{R}\}$ from the posterior. The beam sampler was originally introduced [2] in 2008 as an inference algorithm for the infinite Hidden Markov model (iHMM); although the iPOMDP and iHMM share strong similarities, the beam sampler requires some adaptation to be useful in our case. These details are omitted in [1], and so we have read [2] in detail to adapt the procedure to the iPOMDP. Our adaptations, produced after lengthy discussions, appear in our pseudocode. The beam sampler approach makes use of Stirling numbers (of the first kind), and we have reviewed these to determine how to generate them efficiently; this also appears in the *Appendix*. It is also worth noting that the approach in [1] initiates the inference algorithm with some number M of candidate POMDPs, and then runs the inferential loop in parallel on all M POMDPs to obtain M (distinct) corresponding samples from the posterior distribution on each latent variable in the generative model in Figure 1. We let $\{S^{(m)}, \mathcal{T}^{(m)}, \Omega^{(m)}, \mathcal{R}^{(m)}\}$ denote the sampled latent variables for the mth model, for $m \in \{1, \ldots, M\}$. In order to coalesce the output of all M separate POMDPs, the authors in [4] introduce an importance weight for each model, where the importance weight $w_{t+1}(m)$ at timestep $t+1$ for model m is evaluated recursively as follows:

$$w_{t+1}(m) \propto \left[\sum_{s \in S} \Omega^{(m)}(s, a_t, o_t) b_t^{(m)}(s)\right] \cdot \left[\sum_{s \in S} R^{(m)}(s, a_t, r_t) b_t^{(m)}(s)\right] \cdot w_t(m), \tag{2}$$

where $b_t^{(m)}(s) \in \Delta_{|S|}$ is a probability vector whose sth entry denotes the probability that the agent is in state s at time t, according to the specifications of the mth model. We let $w_t \in [0,1]^M$ denote the vector with mth entry $w_t(m)$. Models that are more consistent with the observed variables in the history h_t are given a higher importance weights.

In order to make use of the BURLAP java code library, we chose to implement action selection with the QMDP heuristic of [3], consistent with the action selection approach in [4]. This approach individually solves each of the M POMDPs. At timestep t, the agent takes the action $a_{t+1} = \max_{a' \in A} \sum_m w_t(m) Q_m(a')$, where $Q_m(a)$ denotes the value of taking action a in model m.

Results

We replicate Figure 3 and Figure 5 from the paper that we chose for replication [1]. Figure 3 from the paper investigates the performance of the algorithm on two iPOMDP benchmark models, termed "lineworld" and "loopworld".

We depict both models in Figure 2. Our replications (which appear as Figure 3 in our paper) show the evolution of the predicted number of states as a function of episode number. We should note that whereas Doshi-Velez has plotted (in black) the evolution of the mean number of inferred iPOMDP states, averaged over 50 repeated trials, we show results from one trial.

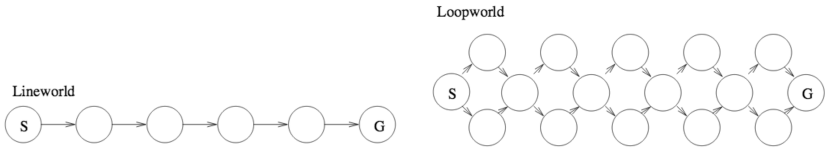


Figure 2: Cartoon of Models

Figure 5 from the paper investigates the performance of the algorithm on the "shuttle" iPOMDP benchmark model. Our replications (which appear as Figure 3 in our paper) show the evolution of total reward, as a function of episode number. It is encouraging that we experience better results than the original paper: this could be due to the fact that our action selection strategy is different from the one that the other used to generate her plot (we discuss this more in the *Discussion*, below).

In the case of both figures, due to the stochastic nature of the algorithm, we do not expect the plots to match exactly. It is interesting, however, that in both cases, our model predicted fewer states.

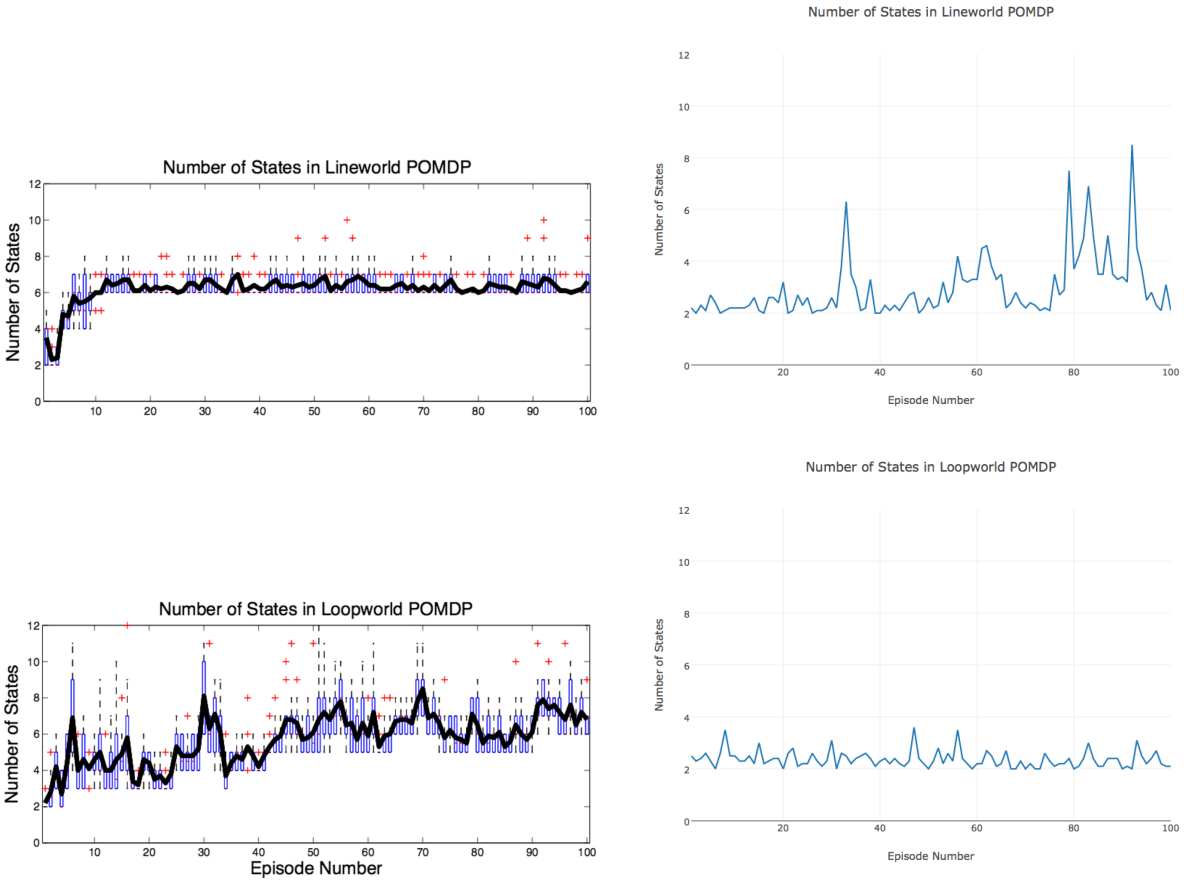


Figure 3: Replication of Figure 3 in [1]

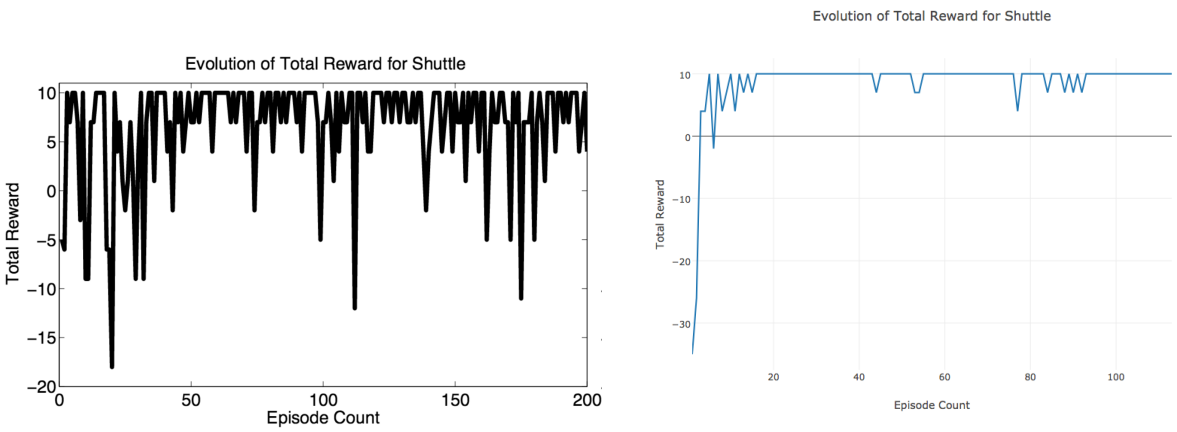


Figure 4: Replication of Figure 5 in [1]

Discussion

In her 2009 paper [1], Doshi-Velez used stochastic forward search with a forward looking tree to perform action selection. However, in her 2015 paper [4], she shows that action selection with the QMDP heuristic has similar performance. This is surprising, since one could reasonably hypothesize that approaches to action selection that take into account the probabilities of future histories may provide added benefit. It is, however, a pleasant surprise, since the QMDP heuristic is more easily implemented. We execute an epsilon-greedy approach which executes the QMDP solution with probability $1 - \epsilon = .9$, and performs a random action with probability ϵ, consistent with the specifications in [4].

The paper that we chose for replication [1] updates importance weights without considering reward; that is, the importance weight $w_{t+1}(m)$ is evaluated recursively as:

$$w_{t+1}(m) \propto \left[\sum_{s \in S} \Omega^{(m)}(s, a_t, o_t) b_t^{(m)}(s) \right] \cdot w_t(m). \tag{3}$$

We decided that the approach for evaluating importance weights in Equation (2) would achieve better performance, since Equation (3) ignores relevant information. That is, rewards should be treated as observed random variables and utilized to update the posterior over POMDP models.

The concentration parameters λ and α used to complete the POMDP experiments in [1] on select problems from the literature (lineworld, loopworld) are never explicitly stated. We have tried multiple values and discovered that their values have minimal impact on performance.

Adaptation and implementation of the beam sampler took the majority of our time for this project. The pseudocode, which appears in the *Appendix*, was drafted from a sequential process involving extensive conversations, acquired intuition, and careful adaptations. The beam sampler involves resampling a large number of latent variables; we were initially not sure if the order in which the variables were sampled matters (specifically, if $\overline{\mathcal{T}}$ must be resampled before or after $\mathcal{T}$). In the end, we decided that it made most sense, in accordance with the idea of message passing from the observed random variables up the latent chain, to sample $\mathcal{T}$ before $\overline{\mathcal{T}}$. We also needed to discuss extensively how to implement the introduction of the auxiliary sequence u for sampling from the potentially countably infinite state space. The original beam sampling paper [2] was developed for the iHMM, which has state-dependent transitions, as opposed to transitions that depend on state-action pairs. Initially, we were unsure about how to handle this added complexity, where we have to deal with a three-dimensional tensor, as opposed to a two-dimensional matrix.

Another idea that we had that is worth mentioning is that we were initially tempted to eliminate the calculation of importance weights from the algorithm. We hypothesized that it would be completely feasible to instead specify a prior and draw a single model that could improved, based on the history, at every timestep. However, after

writing all of the code for the beam sampler, we realized that the algorithm runs quite slowly, where speed decays significantly as the history h_t grows with time t. Thus, to speed the algorithm, it is necessary to use importance weights to maintain a reasonable level of speed. That is, the importance weights allow us to *slightly* amend our action selection strategies between beam sampling steps, based on the history, where the beam sampler represents a more *substantive* inferential amendment to our beliefs of the underlying POMDP. It is still important to note that, although we achieve faster performance with running the beam sampler less often, we still occasionally need to work with sampling a very large latent state space. Although we are not sure, we hypothesize that the author in [1] may occasionally throw away early pieces of the history with the goal of sampling a smaller collection of latent states. However, this was not described in the paper; in our replications, we keep the entire history.

References

[1] F. Doshi-Velez, "The Infinite Partially Observable Markov Decision Process," *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2009.

[2] J. Van Gael, Y. Saatci, Y. Teh, and Z. Ghahramani, "Beam Sampling for the Infinite Hidden Markov Model," in *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[3] M. Littman, A. Cassandra, and L. Kaelbling, "Learning policies for partially observable environments: scaling up," *International Conference in Machine Learning*, 1995.

[4] F. Doshi-Velez, D. Pfau, F. Wood, and N. Roy, "Bayesian Nonparametric Methods for Partially-Observable Reinforcement Learning," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, 2013.

Appendix 1: Pseudocode for Beam Sampler

Algorithm *Sample u*
Input: $\{s_t\}_{t=1}^{T}$, $\{a_t\}_{t=0}^{T-1}$, $\mathcal{T}$
Output: $\{u_t\}_{t=1}^{T}$
1. **for** $t \leftarrow 1$ **to** T
2. **do** $u_t \sim \text{Uniform}(0, T(s_{t-1}, s_t, a_{t-1}))$
3. **return** $u = \{u_t\}_{t=1}^{T}$

Algorithm *Sample s*
Input: Old estimates for $\{s_t\}_{t=1}^{T}$; u, $\{a_t\}_{t=0}^{T-1}$, $\{r_t\}_{t=0}^{T-1}$, $\{o_t\}_{t=1}^{T}$ S, $\mathcal{O}$, $\mathcal{R}$, $\mathcal{T}$
Output: Sampled $\{s_t\}_{t=1}^{T+1}$; up-/down-sampled S, $\mathcal{O}$, $\mathcal{R}$, $\mathcal{T}$
($*$ Note that u_t corresponds to a_{t-1} for all $t \in \{1, \ldots, T\}$ $*$)
1. Let $u_{min} = \min\{u_t : t \in \{1, \ldots, T\}\}$.
2. **for** $j \leftarrow 1$ **to** $|S| \cdot |A|$
3. **do** Stick-break the last entry in the j-th row according to the prior. Store these new values successively by creating new entries in the row. Stop when the stick has remaining size less than u_{min}.
4. The above step will produce a data structure that has variable row size; make the data structure a rectangular matrix by padding shorter rows with zeros. Stick-breaking has also produced new states; extend the matrix length-wise to account for these new states by sampling from the prior distribution. Call the resultant matrix $\mathcal{T}^{(1)}$.
5. Let $\mathcal{T}_{ij}^{(2)}$ be the row of the matrix $\mathcal{T}^{(2)}$ that will correspond to the ith starting state and the jth action. Initialize $\mathcal{T}^{(2)} = \mathcal{T}^{(1)}$.
6. **for** $t \leftarrow 1$ **to** T
7. **do** Set $i = s_{t-1}$, $j = a_{t-1}$.
8. **do** Let $\mathcal{T}_{ij}^{(2)} = \mathcal{T}_{ij}^{(2)} * \mathbb{I}(\mathcal{T}_{ij}^{(2)} > u_t)$, where * denotes elementwise multiplication.
9. Expand $\mathcal{O}$ and $\mathcal{R}$ to account for new states by sampling from the prior.
10. Sample s_{T+1} from the posterior obtained from forward filtering.
11. **for** $t \leftarrow T$ **to** 1
12. **do** Sample s_t from s_{t+1} through a backward pass.

Algorithm *Sample transition matrix*
Input: $\{s_t\}_{t=0}^{T}$, $\{a_t\}_{t=0}^{T-1}$, α, $\overline{\mathcal{T}}$
Output: Resampled $\mathcal{T}$, $n = \{n_{ij} : i \in \{1, \ldots, |S|\}, j \in \{1, \ldots, |A|\}\}$
1. Let $|S|$ be the number of distinct states in $\{s_t\}_{t=1}^{T}$. Let $|A|$ be the number of distinct actions in $\{a_t\}_{t=0}^{T-1}$. The output $\mathcal{T}$ will satisfy $\mathcal{T} \in [0,1]^{(|S|+1)\cdot|A|\times(|S|+1)}$.
2. **for** $i \leftarrow 1$ **to** $|S|$
3. **for** $j \leftarrow 1$ **to** $|A|$
4. **do** let n_{ijk} be the number of times that we started in state i, did action j, and ended up in state k. That is, $n_{ijk} = \sum_{t=0}^{T-1} \mathbb{I}(s_t = i, a_t = j, s_{t+1} = k)$ for all $k \in \{1, \ldots, |S|\}$.
5. **do** let $\mathcal{T}_{ij}$ be the row of $\mathcal{T}$ corresponding to the ith starting state and the jth action. $\mathcal{T}_{ij} \sim \text{Dirichlet}(n_{ij1} + \alpha\overline{\mathcal{T}}(1), \ldots, n_{ij|S|} + \alpha\overline{\mathcal{T}}(|S|), \alpha \sum_{i=|S|+1}^{\infty} \overline{\mathcal{T}}(i))$
6. We now populate the remaining $|A|$ entires of $\mathcal{T}$, corresponding to starting in any of the states that we believe we have not yet visited, and taking the j-th action, for $j \in \{1, \ldots, |A|\}$ **for** $j \leftarrow 1$ **to** $|A|$
7. **do** Let $\mathcal{T}_{(|S|+1)j}$ be the row of $\mathcal{T}$ corresponding to starting in a state we believe we have not yet visited and taking action j.
8. **do** $\mathcal{T}_{(|S|+1)j} \sim \text{DP}(\alpha, \overline{\mathcal{T}})$
9. **return** $\mathcal{T} \in [0,1]^{(|S|+1)\cdot|A|\times(|S|+1)}$

Algorithm *Calculate all Stirling numbers for time T*
Input: Stirling matrix $\mathcal{S} \in (\mathbb{Z}^*)^{T\times T}$
Output: Stirling matrix $\mathcal{S} \in (\mathbb{Z}^*)^{(T+1)\times(T+1)}$
($*$ Note $\mathcal{S}_{0,0}$ corresponds to $\left\{{0 \atop 0}\right\}$, and $\mathcal{S}_{i,j}$ corresponds to $\left\{{i \atop j}\right\}$ $*$)
1. Let $\mathcal{S}_{T,0} = 0$, $\mathcal{S}_{T,T} = 1$
2. **for** $k \leftarrow 1$ **to** $T - 1$
3. **do** $\mathcal{S}_{T,k} = T \cdot \mathcal{S}_{T-1,k} + \mathcal{S}_{T-1,k-1}$
4. **return** $\mathcal{S} \in (\mathbb{Z}^*)^{(T+1)\times(T+1)}$

Algorithm *Sample m*
Input: $\{s_t\}_{t=0}^{T}$, $\overline{\mathcal{T}}$, α, n
Output: $m = \{m_{ijk} : i, k \in 1, \ldots, |S|, j \in 1, \ldots, |A|\}$
1. **for** $i \leftarrow 1$ **to** $|S|$
2. **for** $j \leftarrow 1$ **to** $|A|$
3. **for** $k \leftarrow 1$ **to** $|S|$
4. **do** $S(n_{ijk}, m) \leftarrow \mathcal{S}_{n_{ijk},m}$ (refers to matrix of Stirling numbers calculated in above algorithm)
5. **do** Sample m_{ijk} from the conditional distribution $\mathbb{P}(m_{ijk} = m|n, \overline{\mathcal{T}}, \alpha) \propto S(n_{ijk}, m)(\alpha\overline{\mathcal{T}}(k))^m$.
6. **return** $m = \{m_{ijk} : i, k \in 1, \ldots, |S|, j \in 1, \ldots, |A|\}$

Algorithm *Sample mean transition distribution*
Input: $m = \{m_{ijk} : i, k \in 1, \ldots, |S|, , j \in 1, \ldots, |A|\}$, λ
Output: Resampled $\overline{\mathcal{T}} \in [0, 1]^{|S|}$
1. Let $m_{..k} = \sum_{i=1}^{|S|} \sum_{j=1}^{|A|} m_{ijk}$.
2. Let $\overline{\mathcal{T}} \sim \text{Dirichlet}(\lambda m_{..1}, \ldots, \lambda m_{..|S|})$
3. **return** $\overline{\mathcal{T}} \in [0, 1]^{|S|}$.

Algorithm *Sample observation matrix*
Input: o, O, H
Output: $\mathcal{O} \in \mathbb{R}^{(|S|+1)\cdot|A|\times|O|}$
1. Note: we assume the set of possible observations (O) is known to the agent beforehand.
2. **for** $i \leftarrow 1$ **to** $|S|$
3. **for** $j \leftarrow 1$ **to** $|A|$
4. **do** let σ_{ij} be the $|O|$-dimensional vector that will count observations immediately ensuing from starting in state i, and doing action j. Specifically, we let, $\sigma_{ij}(k) = \sum_{t=0}^{T-1} \mathbb{I}(s_t = i, a_t = j, o_t = O(k))$ for all $k \in \{1, \ldots, |O|\}$.
5. **do** let $\mathcal{O}_{ij}$ be the row of $\mathcal{O}$ corresponding to the ith starting state and the jth action: it will be vector of dimension $|O|$. Set $\mathcal{O}_{ij} \sim \text{Dir}(\sigma_{ij}(1)H(1), \ldots, \sigma_{ij}(|O|)H(|O|))$.
6. We now sample the remaining $|A|$ entries (corresponding to unvisited states) of the vector from the prior.
7. **for** $j \leftarrow 1$ **to** $|A|$
8. $\mathcal{O}_{(|S|+1)j} \sim H$
9. **return** $\mathcal{O}$

Algorithm *Sample reward matrix*
Input: r, R, H_R
Output: $\mathcal{R} \in \mathbb{R}^{(|S|+1)\cdot|A|\times|R|}$
1. Note: we assume the set of possible rewards (R) is known to the agent beforehand.
2. **for** $i \leftarrow 1$ **to** $|S|$
3. **for** $j \leftarrow 1$ **to** $|A|$
4. **do** let ρ_{ij} be the $|R|$-dimensional vector that will count rewards immediately ensuing from starting in state i, and doing action j. Specifically, we let, $\rho_{ij}(k) = \sum_{t=0}^{T-1} \mathbb{I}(s_t = i, a_t = j, r_t = R(k))$ for all $k \in \{1, \ldots, |R|\}$.
5. **do** let $\mathcal{R}_{ij}$ be the row of $\mathcal{R}$ corresponding to the ith starting state and the jth action: it will be a vector of dimension $|R|$. Set $\mathcal{R}_{ij} \sim \text{Dir}(\rho_{ij}(1)H_R(1), \ldots, \rho_{ij}(|R|)H_R(|R|))$.
6. We now sample the remaining $|A|$ entries of the vector (corresponding to unvisited states) from the prior.
7. **for** $j \leftarrow 1$ **to** $|A|$
8. $\mathcal{R}_{(|S|+1)j} \sim H_R$
9. **return** $\mathcal{R}$

Appendix 2: Stirling Numbers of the First Kind

Let $\left\{ {n \atop k} \right\} := |S(n,k)|$, where we note that the authors of the beam sampler paper should have included an absolute value. The Stirling numbers:

- arise as coefficients of the rising factorial:

$$x^{(n)} = x(x+1)\cdots(x+[n-1]) = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} x^k.$$

- can be calculated by the recurrence relation

$$\left\{ {n+1 \atop k} \right\} = n\left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}.$$

Let's calculate some values. There is a chart on Wikipedia of Stirling numbers. We'll check to make sure that we can write code to match the values.

- Let $n = 0$. We let $\left\{ {0 \atop 0} \right\} = 1$.
- Let $n = 1$. The corresponding rising factorial is $x^{(1)} = x = x + 0$. Then we can only consider $k \in \{0, 1\}$, and
  - $\left\{ {1 \atop 0} \right\} = 0$
  - $\left\{ {1 \atop 1} \right\} = 1$
- Let $n = 2$. The corresponding rising factorial is $x^{(2)} = x(x+1) = x^2 + x = x^2 + x + 0$. Then we can only consider $k \in \{0, 1, 2\}$, and
  - $\left\{ {2 \atop 0} \right\} = 0$
  - $\left\{ {2 \atop 1} \right\} = 1$
  - $\left\{ {2 \atop 2} \right\} = 1$
- Let $n = 3$. The corresponding rising factorial is $x^{(3)} = x(x+1)(x+2) = x^3 + 3x^2 + 2x = x^3 + 3x^2 + 2x + 0$. Then we can only consider $k \in \{0, 1, 2, 3\}$, and
  - $\left\{ {3 \atop 0} \right\} = 0$
  - $\left\{ {3 \atop 1} \right\} = 2$
  - $\left\{ {3 \atop 2} \right\} = 3$
  - $\left\{ {3 \atop 3} \right\} = 1$

It can be verified that these numbers match the numbers in the chart on Wikipedia. But we won't program the Stirling numbers in this way. We will instead note that $\left\{ {n \atop 0} \right\} = 0$ and $\left\{ {n \atop n} \right\} = 1$ for all $n \geq 1$. To get the other values for all possible n and k, we will use the recurrence relation.

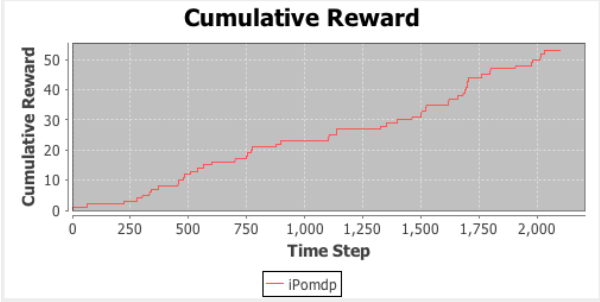
Let $n = 3$. We will use the above recurrence relation to calculate the Stirling numbers corresponding to $n + 1 = 4$, which has potential corresponding values of $k \in \{0, 1, 2, 3, 4\}$. We know $\left\{ {4 \atop 0} \right\} = 0$ and $\left\{ {4 \atop 4} \right\} = 1$, from above. Furthermore, using the recurrence relation,

$$\left\{ {4 \atop 1} \right\} = 3\left\{ {3 \atop 1} \right\} + \left\{ {3 \atop 0} \right\} = 3(2) + 0 = 6$$
$$\left\{ {4 \atop 2} \right\} = 3\left\{ {3 \atop 2} \right\} + \left\{ {3 \atop 1} \right\} = 3(3) + 2 = 11$$
$$\left\{ {4 \atop 3} \right\} = 3\left\{ {3 \atop 3} \right\} + \left\{ {3 \atop 2} \right\} = 3(1) + 3 = 6$$

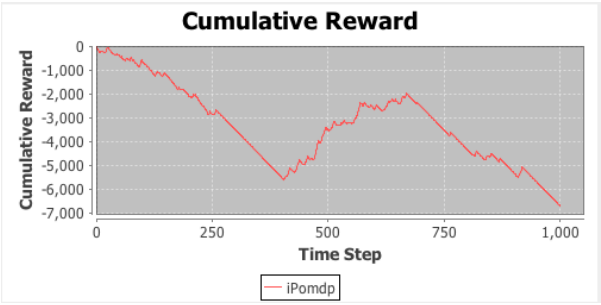
Now we just need to program this. Best practice probably involves storing the Stirling numbers in a large matrix. At each iteration, the values of n that we need to consider correspond to the number of timesteps spent at any given state. But, for some value T, we have $n \leq T + 1$. Thus, we can spread the calculation load evenly over iterations (don't need to compute many Stirling numbers at a single iteration, and can instead expand our matrix of Stirling values by one row at every timestep).

Appendix 3: Other Results

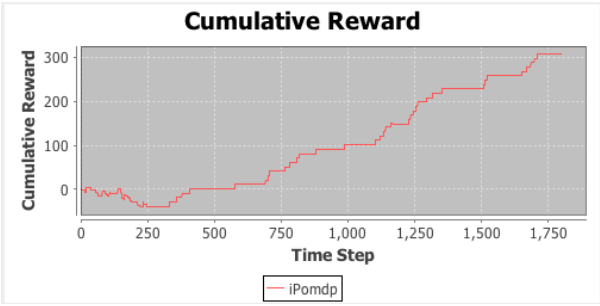
Here, we show results from various other POMDP benchmark models to illustrate potential pitfalls of the algorithm. Note that the algorithm takes random actions until step 250.



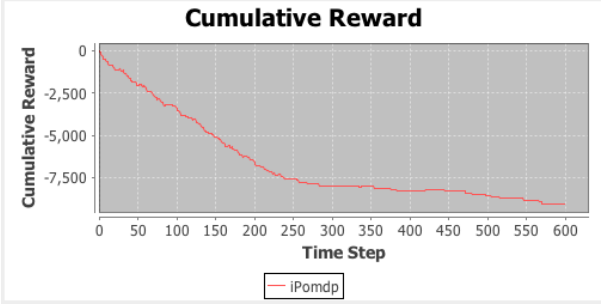
(a) Results for grid POMDP



(b) Results for network POMDP



(c) Results for shuttle POMDP



(d) Results for tiger POMDP

Figure 5: More Results