Study Notes for AI (CS 440/520)
Lecture 18: Partially Observable Markov Decision Processes

Corresponding Book Chapters: 17.4
Note: These notes provide only a short summary and some highlights of the material covered in the corresponding lecture based on notes collected from students. Make sure you check the corresponding chapters. Please report any mistakes in or any other issues with the notes to the instructor.

1 Introducing Partially Observable Markov Decision Processes

In a Markov decision process, it is assumed that the environment is *fully observable.* This means that the agent knows exactly where it is in the state space at all times. This also means that the optimal policy depends only on the current state. This is not the case when the environment is *partially observable.* For such problems, the agent does not know exactly where it is in the state space and, instead, operates on a probability distribution of possible states. An appropriate formulation for such problems is that of Partially Observable Markov Decision Processes (POMDPs). POMDPs are extensions of Markov Decision Processes (MDPs), which also allow uncertainty in observations.

The uncertainty in observations implies that the agent does not know exactly its state. Similarly, the agent is not able just to execute an action $\pi(s)$ for state s, because there is no certainty that this is the correct state of the agent. POMDPs are more difficult problems than MDPs. The objective of a POMDP is exactly the same as an MDP, which is to maximize the expected utility of the agent.

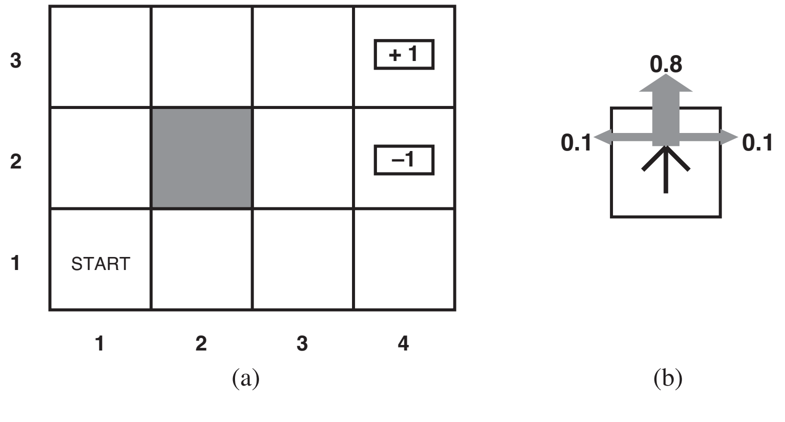


Figure 1: (a) Simple 4x3 world with two goal states. (b) Probabilities of outcomes given intended movement

As an example, consider the world shown in Figure 1. The agent can move in any direction given that there is no wall or obstacle. When the agent moves, there is an 80% chance that the agent will actually move in the intended direction, a 10% chance that the agent will move sideways to the left, and a 10% chance that the agent will move sideways to the right. Let's assume that the agent has *absolutely no sensors* and has *no idea where it is*. Then initially, the agent has an equal probability to be in any of the possible states (excluding goal states) as in Figure 2(a). So what should the agent do in this scenario? One possible answer is that the agent should act in such a say that it reduces its uncertainty, or act in such a way that it has a better idea of where it might be. For example, if the agent attempts to move *Left* five times, then it is much more likely that it is at the left wall (Figure 2(b)). If it then attempts to move *Up* five times, it will most likely be in the top row (Figure 2(c)). Finally, if the agent moves *Right* five times, then there will be about a 77.5% chance that it will reach the +1 goal state (Figure 2(d)). At first glance, this may seem like a decent solution. However, the expected utility would be 0.08 because we assume a -0.04 penalty in each empty cell at the end of each attempted move. An optimal policy will be described later that does much better.

This example also shows that we need a probability distribution function, or a belief distribution, to represent where the agent is inside the environment. Consequently, instead of assigning actions to states, in POMDPs we have to assign actions to belief state distributions.

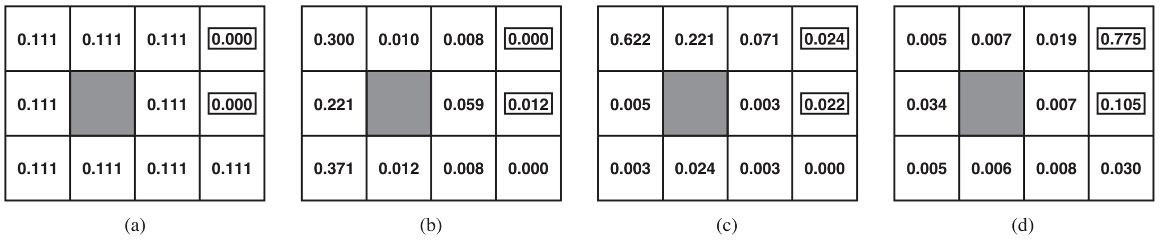


Figure 2: The probability of being at each state after executing the following action sequence: move 5 times left, move 5 times up, moves 5 times right. After applying all these actions, there is a 77.5% that we are able to reach the cell +1. Continuing moving right increases the chances to reach the goal to 81.8% and an expected utility of 0.08. We describe in the text approaches that compute policies which reach the goal with higher probability and better reward.

The definition of a POMDP is very similar to that of an MDP. There still exists a transition model and rewards function, but there is also one additional element - an *observation model* $O(s, o)$ that specifies the probability of perceiving the observation o in state s.

- Initial probability distribution: b_0
- Transition model: $T(s, a, s')$
- Rewards function: $R(s)$
- Observation model: $O(s, o)$

POMDPs employs filtering to compute the current belief distribution given all the available evidences up to this point. Equation 1 can be applied to compute the belief state distribution:

$$b'(s') = \alpha \cdot O(s', o) \cdot \sum_s T(s, a, s') \cdot b(s) \tag{1}$$

Where $b' = Forward(b, a, o)$. So in a POMDP the optimal action relies on the agents belief state distribution and thus we are interested in an optimal policy over belief states ($\pi^*(b)$) than over singular states ($\pi^*(s)$). Once we have an optimal assignment of actions to belief distributions ($\pi^*(b)$), then the answer to a POMDP can be achieved with the following steps:

1. Given the current belief state b, execute the action $a = \pi^*(b)$
2. receive observation o

3. set the current belief state to Forward(b, a, o)
4. repeat

2 Conversion to a Markov Decision Process

How do we compute, however, the optimal policy over belief states? One major complication is that the space of belief state distributions is a continuous space and we cannot possible one action for each possible value of the belief state distribution. This means that even when we are dealing with an apparently discrete world, such as the 4x3 world example, as soon as we have uncertainty in sensing, the problem becomes continuous.

A second complication is the fact that a secondary objective is necessary to be defined. It is often necessary in the context of a POMDP to select an action that does not bring the agent closer to the goal but instead provides information about the state the agent is currently in. Thus, it is important to evaluate an action not only in terms of whether it allows us to solve the problem but also on the amount of information it provides. In some sense, an agent has to solve at each step a version of the value of information problem: Do I try to reach to the goal as fast as possible or do I invest in decreasing my uncertainty about my state?

Given the above discussion, one possible way (exact) to address POMDPs, is to turn them into MDPs that operate over belief distributions instead of singular states. Thus we have to compute the transition model for this new version of an MDP:

$$\begin{aligned}\tau(b,a,b') &= P(b'|a,b)\\ &= \textstyle\sum_o P(b'|o,a,b)\cdot P(o|a,b) \quad \text{(conditioning)}\end{aligned}$$

The second probability is something that we have to deal explicitly. It can be computed as follows:

$$\begin{aligned}P(o|a,b) &= \textstyle\sum_{s'} P(o|a,s',b)\cdot P(s'|a,b) && \text{(conditioning)}\\ &= \textstyle\sum_{s'} O(s',o)P(s'|a,b) && \text{(rewriting observation model)}\\ &= \textstyle\sum_{s'} O(s',o)\sum_s T(s,a,s')b(s) && \text{(conditioning)}\end{aligned}$$

Now that we have an expression of $P(o|a,b)$ that depends on the state-level observation $O(s',o)$ and transition models $T(s,a,s')$, which are available to us as input, we can replace this expression to the computation of the belief-level transition model:

$$\tau(b,a,b') = \sum_o P(b'|o,a,b)\sum_{s'} O(s',o)\sum_s T(s,a,s')b(s)$$

With the above equation we are actually turning the original POMDP problem over the state space into an MDP problem that assigns actions to belief states. The last element that we have to define for this type of MDPs is how is the reward function computed over the belief states:

$$\rho(b) = \sum_s b(s)\cdot R(s) \tag{2}$$

Advanced versions of value iteration and policy iteration can be used to solve such problems, but we do not study such algorithms in this class. Using such an algorithm, the optimal policy for our original example would be as follows:

$[Left, Up, Up, Right, Up, Up, Right, Up, Up, Right, Up, Right, Up, Right, Up, ...]$

Some confusion may arise from this example. Why do we have an observation model when there are no sensors? The problem is deterministic in belief state space; therefore, there are no observations. The “trick” is to have the agent move *Left* once to ensure that it is not in the lower right corner of the world, so that it is fairly safe to move *Up* and to the *Right*. This agent will reach the +1 goal 86.6% of the time, and will have an expected utility of 0.38 - a large improvement over our original 0.08.

POMDPs with observations are very difficult to come up with an approximate optimum policy (PSPACE-hard). It is infeasible even with a few dozen states. The model of decision-theoretical agents we will describe in the next section is an approximate method to deal with the complexity of POMDPs. It is also the most general version of decision-theoretic agents that we will cover in this class.

3 Decision-Theoretic Agents

The agent design for partially observable, stochastic environments has the following elements:

- The transition and observation models are represented by a *dynamic Bayesian network*.
- The dynamic Bayesian network is extended with decision and utility nodes, as used in *decision networks*. The resulting model is called a *dynamic decision network* or DDN.
- A filtering algorithm is used to incorporate each new percept and action and to update the belief state representation as in Equation (1).
- Decisions are made by projecting forward possible action sequences and choosing the best one.

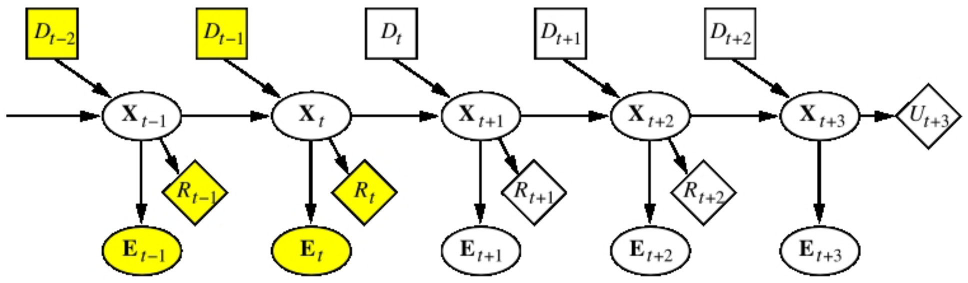


Figure 3: A generic structure of a dynamic decision network.

Decisions are made by:

- Projecting forward possible action sequences
- Estimating their effects
- Choosing the best one

Because we are using dynamic Bayesian networks we go back to the old notation as in Figure 3.

Dynamic Bayesian Network	POMDPs
X_t	s
E_t, A_t	o_t, a_t
$P(X_{t+1}\|X_t, A_t)$	$T(s, a, s')$
$P(E_t\|X_t)$	$O(s, o)$
	R_t (Reward)
	U_t (Utility)

Here we focus on look-ahead methods that estimate the future outcomes and make decisions according to them, as in stochastic game-playing algorithms like EXPECTIMINIMAX for the game of backgammon. The current and future decision, future observations and rewards are all unknown. Notice that X_{t+1} and X_{t+2} have rewards associated with them but X_{t+3} has a utility. This is the expected utility we will receive from that state into the future (after time step $t+3$). In Figure 4 we have the part of the look-ahead solution of the DDN. After an action, an observation occurs. We treat observations like the chance plies we had in game-playing algorithms since they are also probabilistic in nature. A decision can be made by backing up the utility values from the leaves and, taking the average at every observation node and taking the maximum at every decision node. This approach is different from the EXPECTIMINIMAX in the following aspects:

- We have rewards along non-leaf nodes
- Decision nodes correspond to belief states rather than actual states

One optimization on top of look-ahead techniques is to sample from the set of possible observations instead of summing-over all possible observations.

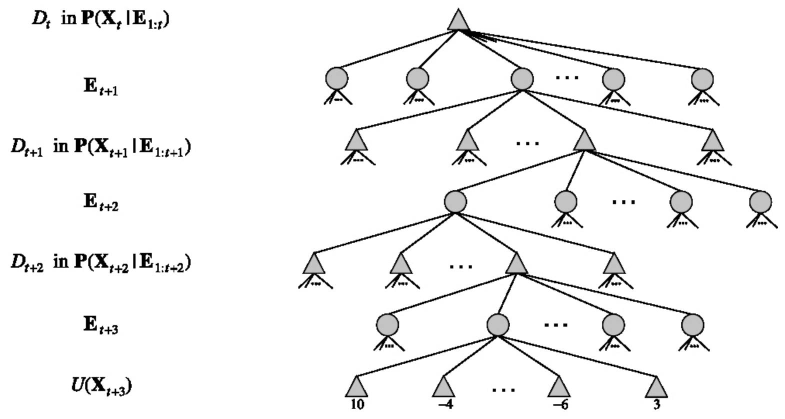
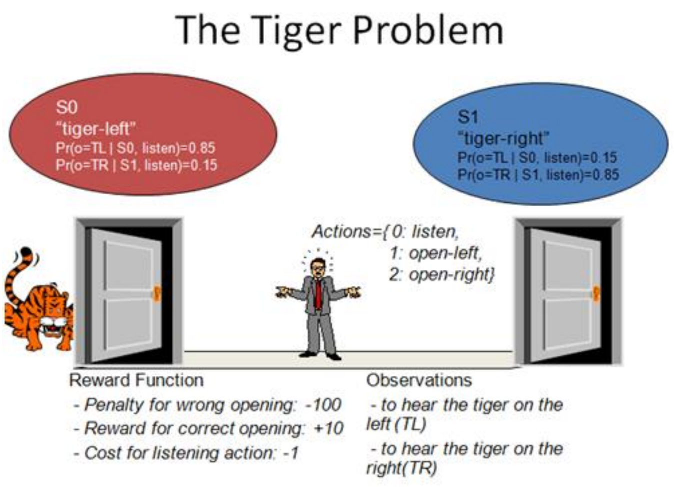
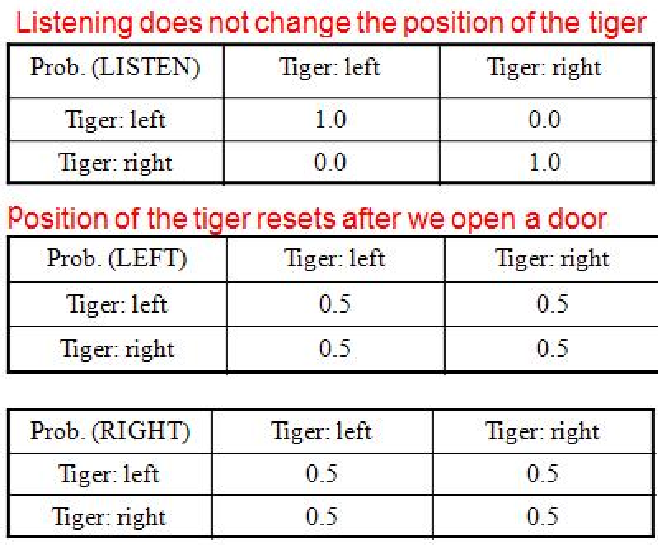

D_t in $\mathbf{P}(\mathbf{X}_t \mid \mathbf{E}_{1:t})$
$\mathbf{E}_{t+1}$
D_{t+1} in $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{E}_{1:t+1})$
$\mathbf{E}_{t+2}$
D_{t+2} in $\mathbf{P}(\mathbf{X}_{t+2} \mid \mathbf{E}_{1:t+2})$
$\mathbf{E}_{t+3}$
$U(\mathbf{X}_{t+3})$
10
–4
–6
3

Figure 4: Part of the look-ahead solution of the DDN

4 Example of a POMDP: The Tiger Problem

The tiger problem is shown below and we can solve this problem using the look ahead method. The objective of this game is to open the door that doesn't have a tiger behind it. The three actions are to open the left door, open the right door, or listen. A reward of 10 is issued if the door has no tiger and -100 if there is. Also a point is deducted for every listening. The tiger is randomly shuffled between doors after each door opening, however the tiger does not move if the player chooses to listen.


The Tiger Problem
S0
"tiger-left"
Pr(o=TL | S0, listen)=0.85
Pr(o=TR | S1, listen)=0.15
S1
"tiger-right"
Pr(o=TL | S0, listen)=0.15
Pr(o=TR | S1, listen)=0.85
Actions={ 0: listen,
1: open-left,
2: open-right}
Reward Function
- Penalty for wrong opening: -100
- Reward for correct opening: +10
- Cost for listening action: -1
Observations
- to hear the tiger on the left (TL)
- to hear the tiger on the right(TR)


Listening does not change the position of the tiger

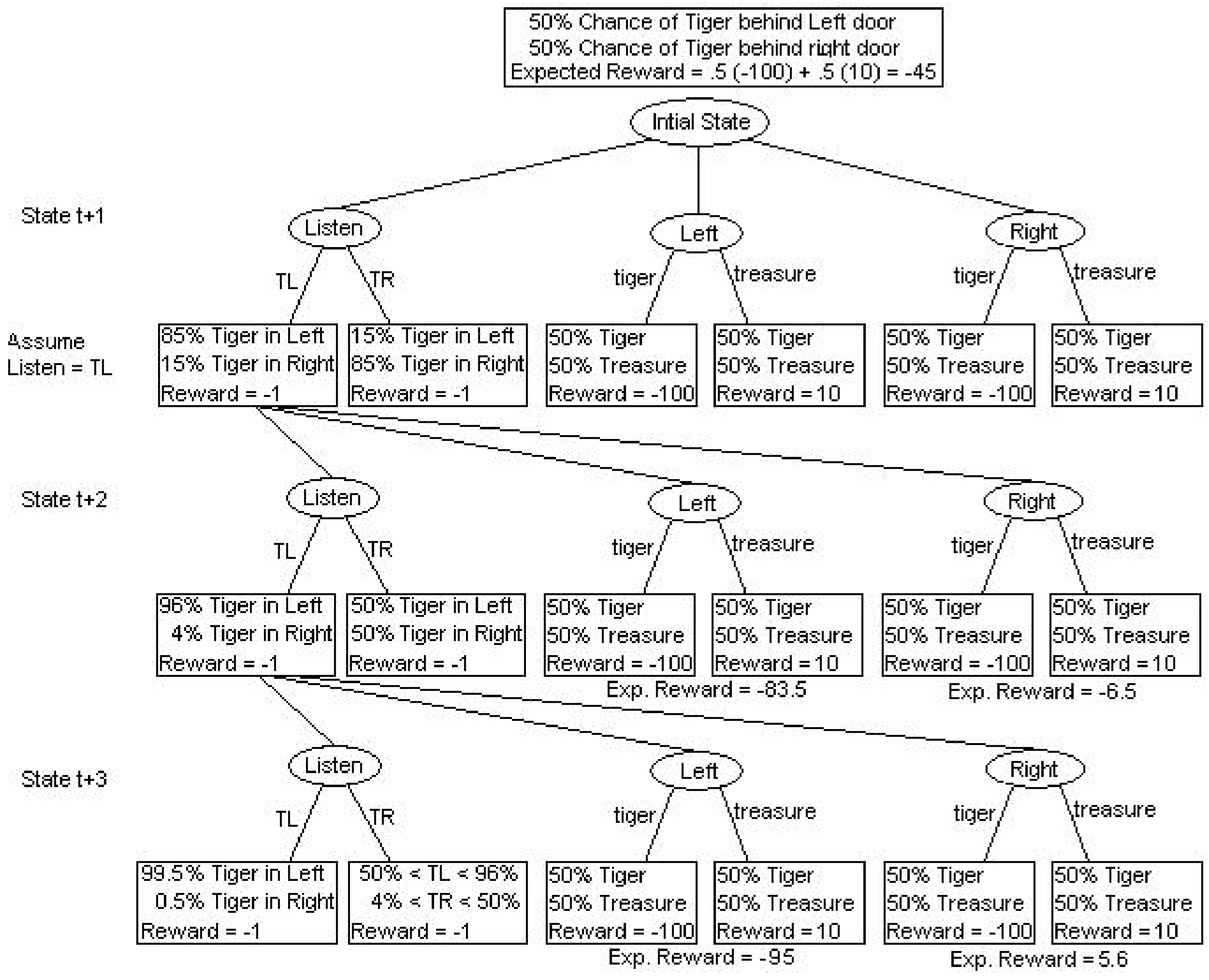
Prob. (LISTEN)	Tiger: left	Tiger: right
Tiger: left	1.0	0.0
Tiger: right	0.0	1.0

Position of the tiger resets after we open a door

Prob. (LEFT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Prob. (RIGHT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Figure 5: Description of the Tiger Problem.



50% Chance of Tiger behind Left door
50% Chance of Tiger behind right door
Expected Reward = .5 (-100) + .5 (10) = -45
Intial State
State t+1
Listen
Left
Right
TL
TR
tiger
treasure
Assume Listen = TL
85% Tiger in Left 15% Tiger in Right Reward = -1
15% Tiger in Left 85% Tiger in Right Reward = -1
50% Tiger 50% Treasure Reward = -100
50% Tiger 50% Treasure Reward = 10
State t+2
96% Tiger in Left 4% Tiger in Right Reward = -1
50% Tiger in Left 50% Tiger in Right Reward = -1
Exp. Reward = -83.5
Exp. Reward = -6.5
State t+3
99.5% Tiger in Left 0.5% Tiger in Right Reward = -1
50% < TL < 96% 4% < TR < 50% Reward = -1
Exp. Reward = -95
Exp. Reward = 5.6

Figure 6: Decision-theoretic Tree for the Tiger Problem.