Paper Review

Acting Optimally in Partial Observable Stochastic Domains
Authors: Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra

Artificial Intelligence - 2
Indian Institute of Technology, jodhpur

Reviewers:
Umang Rajendra Barbhaya (M20CS017)

Date: March 31, 2021

Abstract: A fully Observable environment, in reality, is not possible. An agent has to work with a partially observable stochastic environment only. Whether the environments are virtual or physical authors are trying to find the near-optimal and optimal solution strategies such that agents don't have complete information about the environment. Authors have suggested a partially observable Markov decision process (POMDP) as an update to Markov Decision Process (MDP). Authors have further mentioned that this new algorithm is more efficient than many other existing algorithms and has also executed this algorithm on small problems and has also mentioned the preliminary result for the same. MDP model with a stochastic environment and a goal has the ability to provide the optimal policies and it is very important in the current planning and research on AI. The only problem with the MDP model is that total observability is an obstacle. This problem is solved by POMDP. In POMDP actions are being treated as it is affecting only the environment and agent's state.

Keywords: MDP, POMDP, belief state, optimal policy

Introduction

In reality, getting the complete observable environment is very rare. Some part of the environment is always ignored. So actions of the agents are based on a partially observable environment. So in such an environment agents takes obvious steps wherever possible and try to gain information about the environment and efficiently the goals are achieved.

• Markov Decision Process (MDP)

In MDP the actions of the agent have some probabilistic results and environment direct access is with the agent. MDP is kind of a base for finding the policy which is optimal.

Below is the MDP Tuple

(S, A, T, R)
Where S: Finite set of the states in an environment
A: Finite set of actions
T: Environment state Transition Model
R: Reward Function. Here, S x A is mapped to real numbers

Whenever a Transition happens from state s to s' there is the possibility that it is based on some action a and when this happens a reward is also given to agent which can be positive or negative. It is written as T(s, a, s') and reward as R(s, a). is a policy used to map from S to A which tells us which action we should take in each situation.

• Partial Observability in MDP

Now the MDP discussed earlier has to be modified for adding the Observability part. Here the Observation is finite, . An observation function is mentioned which maps A x S into a probability distribution that is discrete. O(a, s, o) where o is the observation, a is the action and s is the state. But above is not enough since there could be the case where the observation set can be mapped to the state set and it would look like MDP and a poor performance optimal policy would be generated.

A Further modification is done by adding an internal belief state. The environment can be in multiple states and the discrete probability distribution that the environment is in a state is called a belief state defined as B. b(s) is the probability of belief b on state s.

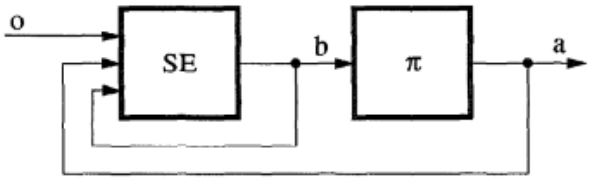

Figure 1: Controller for a POMDP

The above diagram shows how to act in Partial ignorance of the environment. SE stands for state estimator. The last belief state is its input and also the action and the observations which are the most recent and it shares the output as the belief state which is updated. The policy is the other component here whose job is mapping belief state from SE to actions.

T and O are used in constructing the state estimator. Bayes' rule is used over here. The belief state probability vector is the SE's output for every state in the environment. It is represented as below

$$\begin{aligned} \mathrm{SE}_{s'}(b,a,o) &= \Pr(s' \mid a,o,b) \\ &= \frac{\Pr(o \mid s',a,b)\Pr(s' \mid a,b)}{\Pr(o \mid a,b)} \\ &= \frac{O(a,s',o)\sum_{s\in\mathcal{S}} T(s,a,s')b(s)}{\Pr(o \mid a,b)} \end{aligned}$$

Here the normalizing factor Pr(o — a, b) is given as

$$\Pr(o \mid a,b) = \sum_{s'\in\mathcal{S}} O(a,s',o)\sum_{s\in\mathcal{S}} T(s,a,s')b(s) \quad .$$

Consider the example given by the author as below in figure 2,

Given the above 4 states. State 2 is the goal state. One state is always occupied by an agent and it could go left and right one step on a new state. If an end like a wall comes then it does not move and stays in the same state. On reaching the Goal state, moving into any state will do nothing and it will just fetch reward as 1. The problem over here is that it can only observe that it is in the Goal state or not.

Therefore agent cannot know basically what is a true state so it uses the probability vector to represent its belief states. As mentioned above going to any state after the goal state has an equal probability of belief state given as (1/3, 1/3, 0, 1/3). From here new action right is taken then

Figure 2: A Simple POMDP environment

the belief state vector becomes (0, 1/2, 0, 1/2) and state 3 is in which agent is residing. Again if the agent takes a right it hits a wall and it stays in the same state with vector as (0, 0, 0, 1). Deterministic actions here help to reduce the uncertainty.

• Optimal Policies construction

In POMDP it is very difficult to construct the optimal policy. Optimal state-action value function Q*co is a method of MDP that helps to find the optimal policy. Action is generated over here as

$$\text{argmax}_{a \in \mathcal{A}} \sum_s b(s) Q^*_{\text{CO}}(s, a).$$

So basically after some uncertain steps the environment will change from partial observable to completely observable. The number of possible successors |0| is mentioned and based on that the transition function is given as

$$\tau(b, a, b') = \sum_{\{o \in \mathcal{O} | \text{SE}(b,a,o)=b'\}} \Pr(o \mid a, b)$$

Probability is 0 if SE can't generate the new belief. A new reward function based on the R and b is given as below

From above it can be seen that reward is given to the agent for believing it as a good state. Since it is a partially observable environment the agent cannot believe on its own whether a state is good or bad.

Finding this belief is basically a Markov. So the logic over here is that if Optimal policy is picked by an agent for belief MDP then for POMDP also the policy would be optimal. Just one problem which stays is that the belief set is continuous and we need finite state spaces to work in MDP's.

For MDP the optimal policy is found with the help of the Value Iteration. Similarly for POMDP's the optimal policy can be created. Generally, an agent travels the states and then collects the rewards. There are many ways to achieve this, but the author uses the policies that maximize the infinite expected sum of discounted rewards from all states. Basically, the author tries to maximize is a discount factor and r(t) is a time t reward.γ equals zero means increase reward for just one next step. Reward plays an important role when γ increases in the decision process.

Belief state b has an optimal value that is equal to the infinite expected sum of discounted rewards. This happens when the agent starting state is b and the optimal policy is being executed.

$$\rho(b,a) = \sum_{s \in S} b(s) R(s,a)$$

$E[r(0) + \sum_{t=1}^{\infty} \gamma^t r(t)]$, where $0 \leq \gamma < 1$

Below given is the value function

$$V^*(b) = \max_{a \in A} [\rho(b,a) + \gamma \sum_{b' \in B} \tau(b,a,b') V^*(b')]$$

Basically, the value function signifies the current reward plus the next state discounted value which is being generated by taking action that increased this value.

There can also be a policy that can maximize the reward with some finite steps t. t-horizon solutions approach has been used for value iteration since t tends towards infinity. Since t goes to infinity Vt* (Optimal t-horizon policy) goes towards 0 where V* is the difference that is maximum between the optimal infinite-horizon policy's value function.

Above is the value iteration algorithm for the same, this algorithm will definitely converge in the finite number of steps, 2/(1-) is the step limit of optimal policy. Tables are used to represent the finite state MDPs. Some special properties are required in this continuous space to finitely represent this belief MDP. The value function of finite-horizon is always piecewise convex and linear. Convex piecewise-linear function further helps to approximate the value function. The value function of the optimal t-horizon is given below for some Vt sets.

Where Vt is $|S|$-dimensional set of vectors. Similar to Watkins' Q-values Vt can also be viewed as the values associated with different choices in the optimal policy.

• The Witness Algorithm

The aim of this algorithm is of finding the set Vt which is found out using Vt-1. This problem already has a detailed algorithm for the same. Authors have described this new algorithm which is more efficient than the traditional algorithms both theoretically and practically. Similar to other algorithms an approximate value function is constructed over here too, by adding vectors, this equation can be further improved. Using a key insight this set is built for Vt. The alpha can be easily be determined using the belief set b. To add into the vector we have to add the alpha value to the belief set. Now the problem here is we need to find Vt(b) is not equal to the Vt*(b). We need to find the b such that Vt(b) is not equal to the Vt*(b) or else we need to prove that b doesn't exist for such cases.

A linear program is being defined for the witness algorithm from which we get a witness which is a single point such that Vt is not equal to Vt*. The beginning of the process is initially done by populating Vt with the help of vectors from which value function can be represented at the belief space corners.

Let $V_0(b) = 0$ for all $b \in \mathcal{B}$
Let $t = 0$
Loop
  $t := t + 1$
  For all $b \in \mathcal{B}$
    $V_t(b) = \max_{a \in \mathcal{A}} [\rho(b, a)$
      $+\gamma \sum_{b' \in \mathcal{B}} \tau(b, a, b') V_{t-1}(b')]$
Until $|V_t(b) - V_{t-1}(b)| < \epsilon$ for all $b \in \mathcal{B}$

$$V_t(b) = \max_{\alpha \in \mathcal{V}_t} b \cdot \alpha$$

Constraints and variables are defined as below

The last condition says that v' is not equal to v. So we get don't get the witness or we get. It Vt=Vt* then we get a witness and this value is added into the vector and the process is looped again. For a particular vector, we just have only one Vt.

Further for the effective functioning of the algorithm, we need to define a tolerance factor too. Which is represented as . Because of this, the algorithm can stop even when Vt* is not equal to Vt and also the difference between them is less than . This is the factor that makes the witness algorithm more efficient than other algorithms. Approximations are the strategy used in the witness algorithm. With the help of these approximations and value iteration, we can get optimal policies, the only thing we need to do is to keep small. But this is a challenging task as programmatically small values become numerically unstable.

PSPACE-complete is the complexity required for the searching of an optimal policy in the POMDP finite horizon. Also, the algorithms mentioned above have exponential time complexity and an exponential count of vectors is also required for storing the Vt*. But if the case comes that the vectors don't require exponential time then the Witness algorithm will also take less time. Practically witness algorithm has proved very efficient in the terms of running time and also in comparison to the POMDP the witness algorithm can run on very big problems.

• Representing the Policies

When the convergence of the value iteration algorithm takes place then we just have the vector set. As for the optimal value function V* we just have the V-final which constitutes it. Belief space is associated with all the belief vectors which maximize its dot product. Because of which we get the partition of the belief space. Whenever a partition is being shared with the state vector we can have an optimal action. We have a policy set pairs for all the V-final.

Optimal policies are elegantly represented because of the properties of the partitions. Observation and an optimal policy we can transform the belief states in one partition to the belief state of another partition. The policy graph summarizes the choices of the action of the optimal policy.

• Using a Policy Graph

$$\hat{V}_t(b) = \max_{\alpha \in \hat{\mathcal{V}}_t} b \cdot \alpha$$

$$v = V_t^*(b) = \max_{a \in \mathcal{A}} [\rho(b,a) + \gamma \sum_{b' \in \mathcal{B}} \tau(b,a,b') \max_{\alpha \in \mathcal{V}_{t-1}} \alpha \cdot b']$$

and

$$\hat{v} = \hat{V}_t(b) = \max_{\hat{\alpha} \in \hat{\mathcal{V}}_t} \hat{\alpha} \cdot b$$

The optimal policy is represented using the policy graph. With the help of the start node, action is chosen by an agent which also takes into consideration the observations. Correct transitions are made by an agent by seeing observations from the start node. An arc is followed before the next observations and so on. This is the ideal representation of the policy. Agent's past experience and future decisions are summarized from the current node. Arcs help in finding how the observations can be incorporated into the agent which will help to make the decisions. The policy graph gives us a series of steps that we need to execute until we get a goal. From here we can see that this is the optimal strategy used over here. Also, State Estimator is of no use since the policy graph contains all the information an agent can take optimal actions on that

Discussion

Authors have examined various algorithms for solving the POMDP's. Different algorithms have been implemented by authors like the Witness algorithm and a heuristic method too. Based on this algorithm a policy graph is constructed from Vfinal. They tried to restrict the graph less than or equal to the 30 nodes. Normally these graph sizes can become very big. An example problem they have taken had 23 states, 11 observations, and 4 actions and it took less than 30 minutes to converge the algorithm on a policy graph.

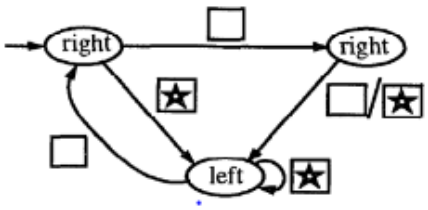

Figure 3: Sample policy graph for the simple POMDP environment

In the above policy graph, the agent found that the belief state and nodes had one-to-one correspondence. Belief state decisions of some environments are grouped together and stored in some nodes. This is basically called generalization.

Above is a 4 x 4 grid where all cells are the same except the Goal state marked with a star. Here only two nodes are present in the optimal policy environment. One node is for moving downwards and the other one is to move towards the right. Kind of step patterns like down and right is taken

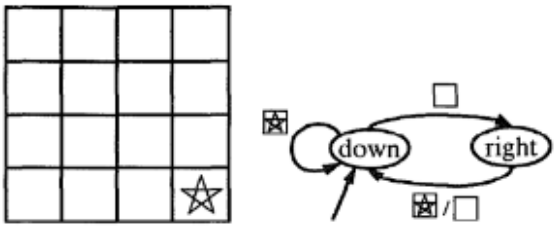

Figure 4: Small unobservable grid and its policy graph

by the agent to move towards the goal state. Once it reaches the goal state, it is again repeated from start. Different beliefs about the environment state the agent takes whenever it is in the down node.

In partially observable environment Agents needs to gain information and this is basically done for more informed decision and improve performance. This phase is generally done as a separate step and not with the environment state-changing actions. But for simplicity Uniform treatment should be given.

Information Gathering phase the authors have modified a classic problem of a tiger behind one door and reward behind the other. There are two doors and Agent's aim is to choose any one of them such that the tiger stays behind one door or a large penalty can be considered and rewards behind the other door. By paying a small penalty we are allowed to listen by going near to the door. If a tiger is behind any door the probability of sound coming from that door will be 0.85 else it will be 0.15.

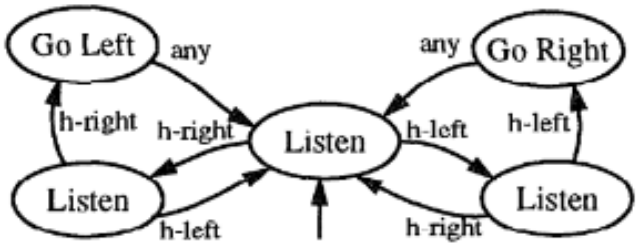

Figure 5: Policy graph for the tiger problem

Now the question comes what is the minimum time anyone should wait before choosing the door. The solution given by the author is as given in the figure above. According to the witness algorithm, we need to stand in the center node. Then check where we can hear the tiger if left then go to the right listen to node, and if we hear tiger on the right then go to the center. Basically, find the node where we hear the tiger sound more and go away from that door. This needs to be done when we hear the tiger sound twice more than the other side. Witness algorithm helps us to achieve a strategy that says that before choosing any state we need to listen which is better than the strategy which doesn't listen.

There is much-related work on POMDPS by Lovejoy 1991 and Monahan 1982. Reinforcement Learning researchers use this strategy a lot. Whitehead Ballard 1991 helps to solve the partial observability problem by taking into consideration some new perceptual data. McCallum 1993 and Chrisman 1992 induce POMDP by environmental interaction and optimal value function is generated by simple approximations. Other work includes TAN 1991 which gets low-cost identification of objects by incorporating decision trees.

The work that Author has presented is preliminary and it can be modified further to perform policy iteration. With the help of stochastic dynamic programming, the author tries to further expand this work and also use function approximation for the optimal value function. This will help to create more good optimal policies. Further POMDP can be used to find an approximate optimal policy by taking into consideration small search space regions.

Conclusion

Authors have experimented with many different algorithms to achieve a solution for POMDPs. With the help of POMDP model, authors are trying to find near-optimal strategies. A large number of tests have been done by the author and they found out that the witness algorithm and heuristic method for policy graph construction came out to be very efficient. A further large number of researches are still ongoing on the POMDPs and also there are many future scope the author is trying to achieve as an extension to this project.

References

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

[2] Astrom, K. J. 1965. Optimal control of markov decision processes with incomplete state estimation. J.Math. Anal. Appl. 10:174-205.

[3] Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1991. Real-time learning and control using asynchronous dynamic programming. Technical Report 91-57, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts.

[4] Bellman, R. 1957. Dynamic Programming. Princeton, New Jersey: Princeton University Press.

[5] Cassandra, A. R.; Kaelbling, L. P.; and Littman, M. L. 1994. Algorithms for partially observable markov decision processes. Technical Report 94-14, Brown University, Providence, Rhode Island.

[6] Cheng, H.-T. 1988. Algorithms for Partially Observable Murkov Decision Processes. Ph.D. Dissertation, University of British Columbia, British Columbia, Canada.

[7] Chrisman, L. 1992. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In Proceedings of the Tenth National Conference on Artificial Intelligence, 183-188. San Jose,California: AAAI Press.

[8] Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In Proceedings of the Eleventh National

[9] Conference on Artificial Intelligence. Howard, R. A. 1960. Dynamic Programming and Murkov Processes. Cambridge, Massachusetts: The MIT Press.

[10] Lovejoy, W. S. 1991. A survey of algorithmic methods for partially observed markov decision processes. Annals of Operations Research 28(1):47-65.

[11] McCallum, R. A. 1993. Overcoming incomplete perception with utile distinction memory. In Proceedings of the Tenth International Conference on Machine Learning. Amherst, Massachusetts: Morgan Kaufmann.

[12] Monahan, G. E. 1982. A survey of partially observable markov decision processes: Theory, models, and algorithms. Management Science 28(l):l-16.

[13] Moore, R. C. 1985. A formal theory of knowledge and action. In Hobbs, J. R., and Moore, R. C., eds., Formal Theories of the Commonsense World. Norwood, New Jersey: Ablex Publishing Company.

[14] Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of markov decision processes. Muthemutits of Operations Research 12(3):441-450.

[15] Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable markov processes over a finite horizon. Operations Research 21:1071-1088.

[16] Sondik, E. J. 1971. The Optimal Control of Purtially Observable Murkov Processes. Ph.D. Dissertation, Stanford University, Stanford, California.

[17] Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Proceedings of the Seventh International Conference on Machine Learning. Austin, Texas: Morgan Kaufmann.

[18] Tan, M. 1991. Cost-sensitive reinforcement learning for adaptive classification and control. In Proceedings of the Ninth National Conference on Artificial Intelligence.

[19] Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. Machine Learning 8(3):279-292.

[20] Whitehead, S. D., and Ballard, D. H. 1991. Learning to perceive and act by trial and error. Machine Learning 7(1):45-83.