

POMDPs and the Witness Algorithm

Planning and acting in partially observable stochastic domains
LP Kaelbling, ML Littman and AR Cassandra, 1998

Zeyn Saigol

IR Lab, School of Computer Science, University of Birmingham

March 4, 2008

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

- (Partially Observable) Markov Decision Processes provide a framework for solving problems in planning under uncertainty.
- Example: Mobile robot navigating in an indoor environment. Results of movement commands are noisy, locations cannot be distinguished perfectly.
- Matches many real-world problems.
- Short presentation...

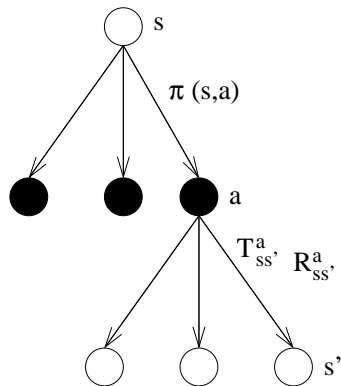
- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

Markov Decision Processes

MDPs are used when actions have stochastic outcomes. An MDP is defined by a tuple $\langle S, A, T, R \rangle$ consisting of

- S , the states of the world
- A , the available actions
- a transition function
 $T(s, a, s') = P(s'|a, s)$
- a reward function $R(s, a)$ for taking action a from state s .

Markov property \Rightarrow future independent of past given s



- The solution to an MDP is a policy $\pi(s)$ that maps states to actions, such that the expectation of the total discounted reward is maximised

$$E \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- The expected reward from starting in a state and then following the optimal policy defines the value of that state:

$$\begin{aligned} V(s) &= \text{ImmediateReward} + \gamma * \text{ExpectedValueOfFutureStates} \\ &= R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s') \end{aligned}$$

- Given the value function, the policy is found by selecting the action that maximises immediate reward plus expected future reward, again by summing over possible successor states.
- The value function can be found using *Value Iteration*.

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

Partial Observability

- Often the agent does not have perfect and complete sensing of the state of the world. Treating observations as the true state is not optimal – same action for all locations that look alike.
- Instead, need to “use memory of previous actions and observations to aid in the disambiguation of the states of the world.”
- POMDPs are a formal method for doing this. Treat actions to gather information and actions to gain reward in a uniform manor, so enable optimal behaviour.
- Defined by $\langle S, A, T, R, \Omega, O \rangle$, with Ω a set of possible observations, and O an observation model
$$O(s', a, o) = P(o|s', a)$$
- The agent's estimate of its current state is best represented as a probability distribution over all possible states, termed the belief state b . This contains all relevant information from past observations, so the Markov property can be preserved.

- State Estimator SE calculates the new belief state according to

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

(from applying Bayes' rule and the Markov property)

As the belief state represents a sufficient statistic for choosing optimal actions, POMDPs can be converted into an MDP in “belief-space”:

- A *continuous* state space β
- Reward function $\rho(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$
- Transition function $\tau(b, a, b') = \sum_{o \in \Omega} P(b'|a, b, o) P(o|a, b)$,

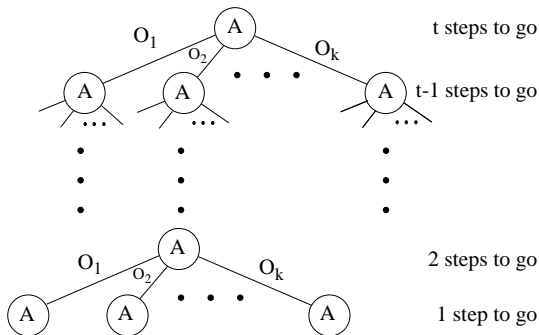
$$\text{where } P(b'|a, b, o) = \begin{cases} 1 & \text{if } SE(b, a, o) = b' \\ 0 & \text{otherwise} \end{cases}$$

The POMDP can then be solved by finding the value function $V(b)$ for this continuous-space MDP.

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

Policy Trees

- Consider finite-horizon POMDPs: Agent has a limited number of timesteps available to it.
- Therefore, optimal policy will be non-stationary.
- Policy trees represent a complete t-step policy for a POMDP.



For different starting belief states, a different policy tree will be optimal.

POMDP value function for a given state and fixed policy tree:

$$\begin{aligned} V_p(s) &= \text{ImmediateReward} + \gamma * \text{ExpectedValueOfFutureStates} \\ &= R(s, a(p)) + \gamma \sum_{s' \in S} T(s, a(p), s') \sum_{o_i \in \Omega} O(s', a(p), o_i) V_{o_i(p)}(s') \end{aligned}$$

where $a(p)$ is the action at the root node of p , and $o_i(p)$ is the $(t-1)$ -step subtree after observing o_i at the root of p .

If define $\alpha_p = \langle V_p(s_1), \dots, V_p(s_n) \rangle$, we find $V_p(b) = b \cdot \alpha_p$.

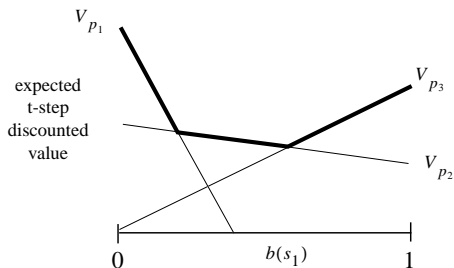
Also,

$$V_t(b) = \max_{p \in \mathcal{P}} b \cdot \alpha_p$$

where \mathcal{P} is the set of all t -step policy trees.

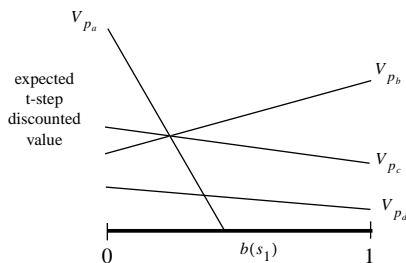
Value Function Properties

2-state example:
($b(s_1)$ defines the belief state)



- The optimal value function is piecewise-linear and convex.
- This means we can project the value function onto the belief space to find which policy tree should be followed for each region of the belief space.

- Most policy trees are dominated by others.
- Can represent value function V_t by a parsimonious set of policy trees \mathcal{V}_t .



- Linear programming can find a point where a vector α corresponding to a given policy tree dominates all other vectors, or prove there are no such points.
- Exhaustive enumeration algorithm:
 - Start with \mathcal{V}_{t-1} , parsimonious set of $(t-1)$ -step policy trees.
 - Build \mathcal{V}_t^+ with all possible combinations of root action, observation, and subtree from \mathcal{V}_{t-1} .
 - Prune policy trees that are not dominant from \mathcal{V}_t^+ to get \mathcal{V}_t .
- The size of \mathcal{V}_t^+ is $|A| |\mathcal{V}_{t-1}|^{|\Omega|} \Rightarrow$ exponential complexity.

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

The Witness Algorithm

- Objective: instead of finding \mathcal{V}_t^+ and pruning to get \mathcal{V}_t , try to generate the elements of \mathcal{V}_t directly.
- Use the concept of Q-values: $Q_t^a(b)$ is the value of taking action a from belief state b and then following the optimal $(t-1)$ -step policy

$$Q_t^a(b) = \sum_{s \in S} b(s)R(s, a) + \gamma \sum_{o \in \Omega} P(o|a, b)V_{t-1}(b')$$

where $b' = SE(b, a, o)$.

- As with the value function, $Q_t^a(b)$ can be represented by a set of policy trees \mathcal{Q}_t^a .
- Witness outer loop at each timestep:

For each action a

$$\mathcal{Q}_t^a = \text{witness}(\mathcal{V}_{t-1}, a)$$

$$\mathcal{V}_t = \text{prune}(\bigcup_a \mathcal{Q}_t^a)$$

The inner loop of the algorithm finds a parsimonious set of policy trees \mathcal{Q}_t^a to represent Q_t^a for a given action, a . It does this by iteratively building up a set of useful policy trees, U_a . At each iteration, it:

- Looks for a witness point b — a point in the belief space where the true $Q_t^a(b)$ differs from the estimate represented by U_a .
- If a witness point is found, adds a new policy tree to U_a that is optimal at that point.
- The root action of the optimal tree is given by the action a .
- For each observation o , the $(t-1)$ -step subtree from \mathcal{V}_{t-1} that is optimal for b' is used, where $b' = SE(b, a, o)$.

Finding Witness Points





for all trees p in U_a
 for all observations o in Ω
 for all trees p' in \mathcal{V}_{t-1}
 create a test policy tree p_{new} from p by
 using p' as the subtree at o
 if find a b such that
 $V_{p_{new}}(b) > V_{\tilde{p}}(b)$ for all $\tilde{p} \in U_a$
 then b is a witness point

- The *witness theorem* specifies that the condition above is sufficient to prove $Q_t^a(b)$ differs from the estimate represented by U_a , and that the estimate is correct if no witness points are found.
- Linear programming can be used to find a witness point given p_{new} and U_a , as the value function is linear in b .

- 1 Introduction
 - Motivation
 - MDPs
- 2 POMDPs
 - POMDP Framework
 - POMDP Value Functions
- 3 Witness
 - The Algorithm
- 4 Paper Impact
 - Legacy and Contribution

Legacy of Littman and Cassandra's Work

- Brought POMDPs to the attention of the AI community (previous work was in operations research, e.g. Smallwood and Sondik, 1973).
- Created the first usable algorithm for solving POMDPs. Witness is still exponential in the worst case, but is polynomial for almost all realistic sample problems.
- The journal article (Kaelbling et al. 1998) has 692 citations according to Google Scholar.
- Inspired work by Nevin Zhang, Joelle Pineau, Sebastian Thrun, Pascal Poupart, Craig Boutilier and many others on developing more efficient algorithms for solving POMDPs (future talks).

-  Kaelbling, L. P., Littman, M. L., and Cassandra, A. R.
Planning and acting in partially observable stochastic domains.
Artificial Intelligence. 101 (1-2):99–134, 1998.
-  Smallwood, R. D. and Sondik, E. J.
The optimal control of partially observable Markov processes
over a finite horizon.
Operations Research. 21(21):1071–1088, 1973.
-  Sutton, R. S. and Barto, A. G.
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, Massachusetts, 1998.
-  Cassandra, A. R.
The POMDP Tutorial.
<http://www.cassandra.org/pomdp/tutorial/>.