COL864: Special Topics in AI
Semester I, 2023-24

POMDPs

Rohan Paul

Outline

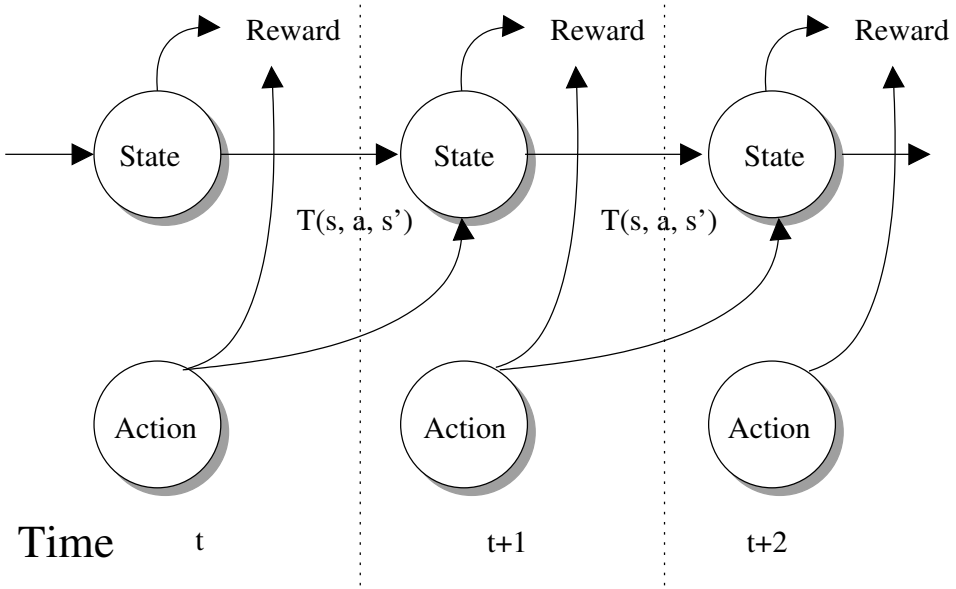
- Last Class
  - Model Based RL (Monte Carlo Tree Search)
- This Class
  - Partially-Observable Markov Decision Processes
- Reference Material
  - Please follow the notes as the primary reference on this topic.
  - Planning and acting in partially observable stochastic domains, *Leslie Pack Kaelbling, Michael L. Littman, Anthony R. Cassandra*, Artificial Intelligence Journal (Sec. 3 and 4 (till 4.1)).
    - https://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf
  - Probabilistic Robotics Chapter on POMDPs
  - AIMA: Ch 17 (Sections 17.4.1 - 17.4.2)

Acknowledgements

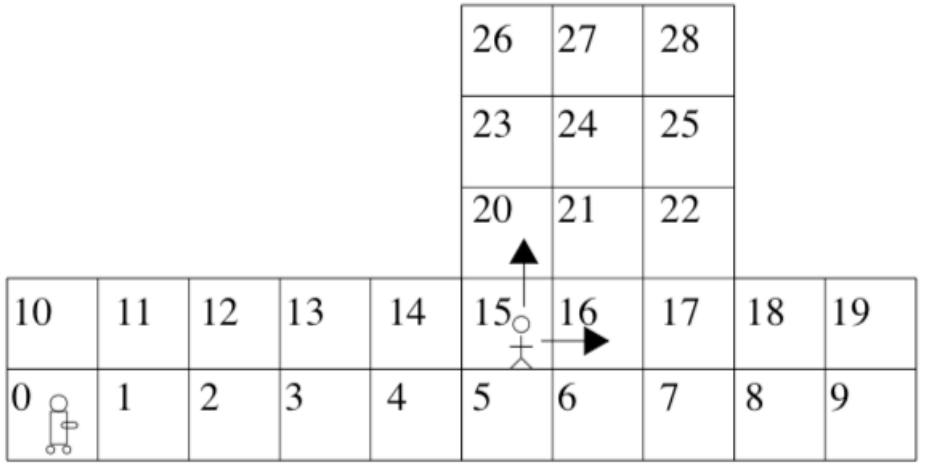
These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Nicholas Roy, Wolfram Burgard, Dieter Fox, Sebastian Thrun, Siddharth Srinivasa, Dan Klein, Pieter Abbeel, Max Likhachev, Alexander Amini (MIT Introduction to Deep Learning) and others.

Expressing MDPs as Graphical Model

- The policy (or controller) chooses the actions that cause the state transitions
- Posterior state is chosen according to the transition function $T(s, a, s')$.
- Structure of the graphical model also makes explicit the Markov assumption ⇒ future states are conditionally independent of past states given knowledge of the current state.
- The Markov assumption models the conditional independence of future states from past states given knowledge of the current state.



Partial Observability: Finding a person



Nursebot Project (CMU). The robot is tasked to locate an elderly person on the floor and deliver them a medicine reminder. The robot can localize itself well but can only know the presence of the person when it is within a range of 2m.

Pineau et al. https://arxiv.org/pdf/1110.0027.pdf

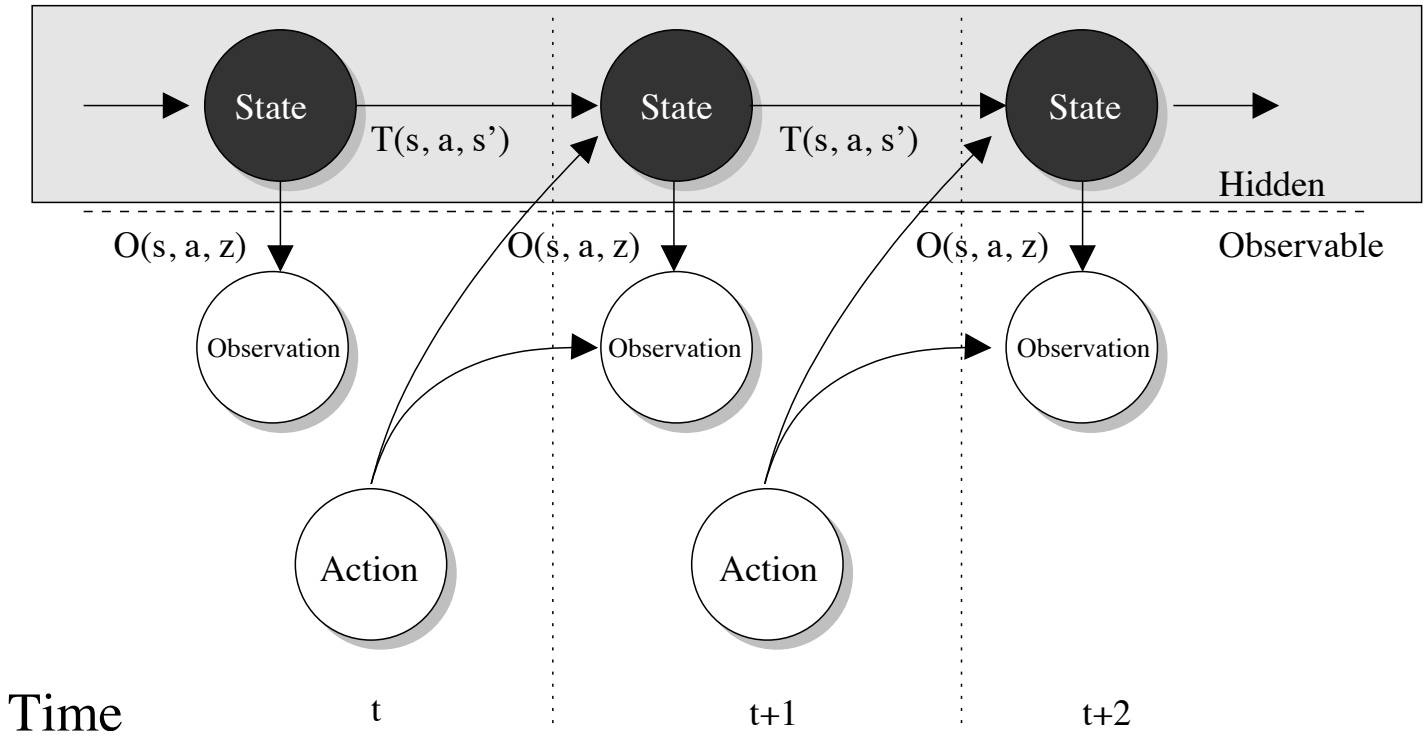
Partial Observability in the MDP Formulation

- State representation: *RobotPosition* and *PersonPosition* (assume discretization of the state space)
- Robot actions: *MoveNorth, MoveSouth, MoveEast, MoveWest* and *DeliverMessage* (when the person and the robot is at the same location)
- Person's motion is stochastic.
- Robot knows its own position but has no knowledge of the person unless the person is within the range of 2m.
- Reward model. R = -1 for any motion action. R = 10 when the robot decides to *DeliverMessage* and the person is in the same cell. Episode ends when the message is delivered.
- Discount factor is 0.95

POMDP Representation

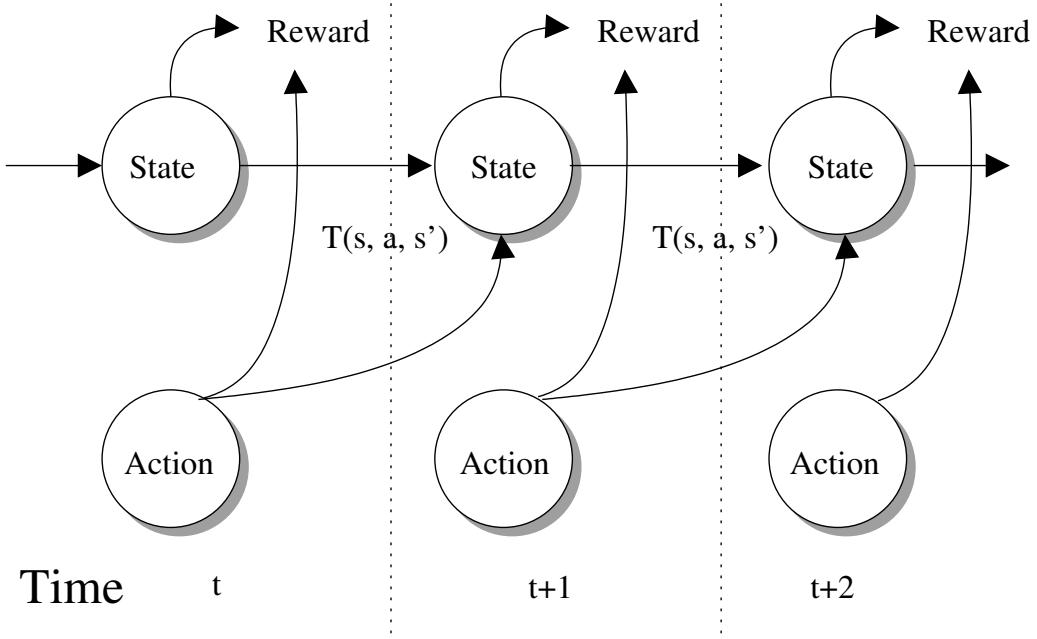
- When the true state of the world cannot be observed directly (or is "inaccessible"), the world is considered as "Partially Observable."
- In partially observable domains, decisions cannot be made based on the state of the world;
- In partially observable domains, decisions are made based on our observations of the world.
- Markov decision process is extended to partially observable Markov decision processes (POMDPs) by adding observations and a probabilistic model of how observations are generated.
- The POMDP is described formally as:
  - a set of states $\mathcal{S} = \{s^1, s^2, \ldots s^n\}$
  - a set of actions $\mathcal{A} = \{a^1, a^2, \ldots, a^m\}$
  - a set of transition probabilities $T(s^i, a, s^j) = p(s^j | s^i, a)$
  - a set of observations $\mathcal{Z} = \{z^1, z^2, \ldots, z^l\}$
  - a set of observation probabilities $O(s^i, a^j, z^k) = p(z \,|\, s, a)$
  - an initial distribution over states, $p(s^0)$
  - a set of rewards $R : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \times S \mapsto \Re$.
  - The reward function for a POMDP is often specified more compactly, such as $R : \mathcal{S} \times \mathcal{A} \mapsto \Re$,

POMDP Graphical Model

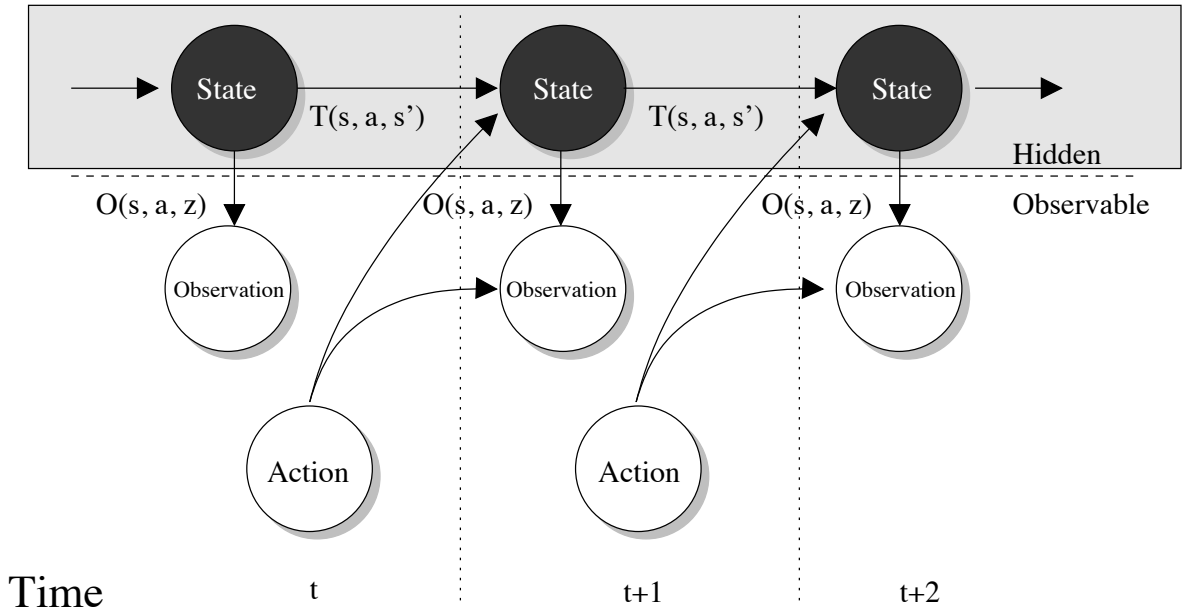


MDPs and POMDPs

MDP Graphical Model



POMDP Graphical Model



Reward not shown for brevity

MDPs and POMDPs

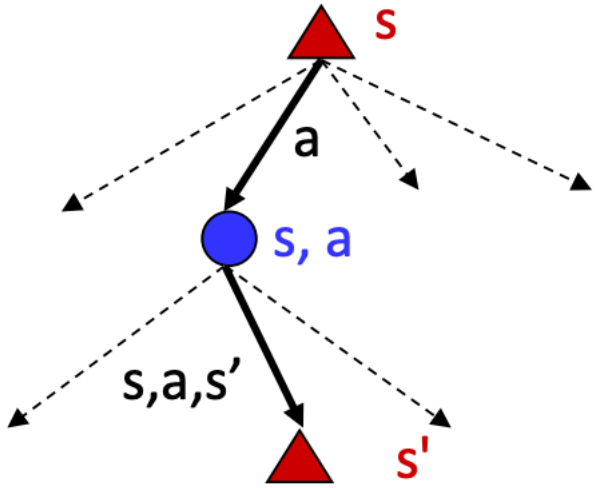
MDPs have:

States S

Actions A

Transition Function $P(s'|s,a)$

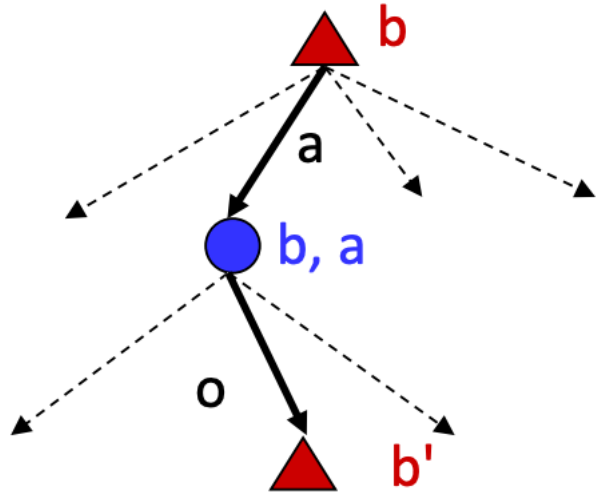
Reward $R(s,a,s')$



POMDPs add:

Observations O

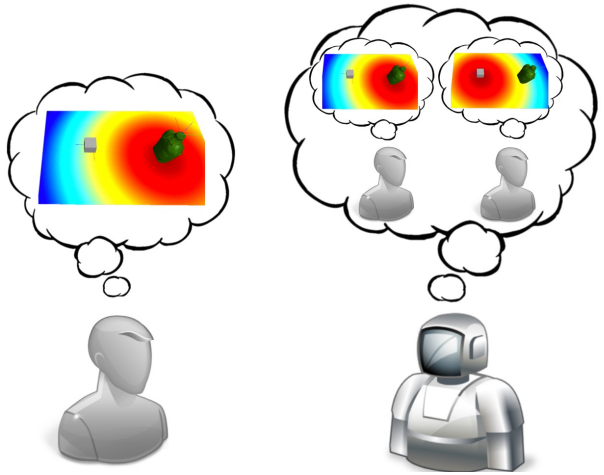
Observation Function $P(o|s)$



Modeling Intent using POMDPs

Human intent is “mostly” unobserved. Dialogue (or gestures) is a way to gather information over the latent belief.

Fig. 1. Demonstration of our FETCH-POMDP model correctly fetching item for user. Note the robot's understanding of implicit information between panels three and four. This reasoning is not hard-coded into our system, but emerges from the solution of our POMDP.



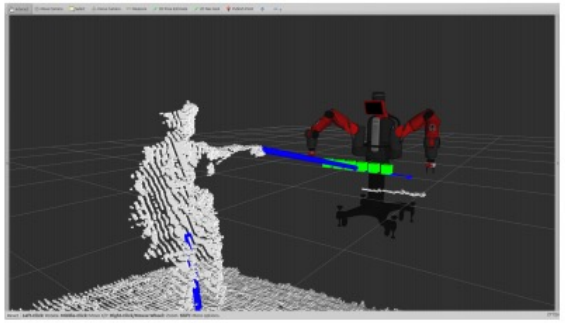


Fig. 3. Screen-capture of RViz visualization of user pointing at item. The blue vectors represent the calculated pointing vectors from each arm. The left arm is down at the user's side, and the right arm is pointing at item four.

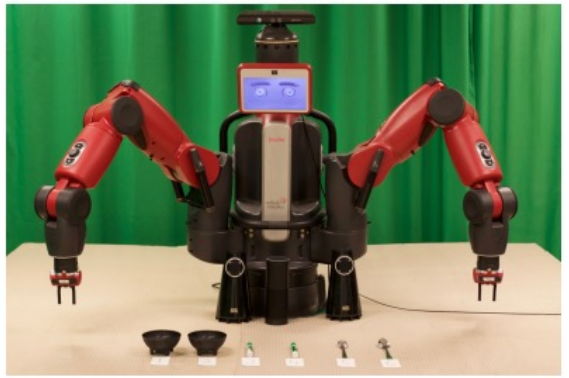


Fig. 4. User's view of robot, with items arranged in the ambiguous configuration.

https://h2r.cs.brown.edu/wp-content/uploads/2017/03/whitney17.pdf

POMDPs for Grasping

Limited sensing of the environment leads to partial observability. Leads naturally to the need for information gathering actions.

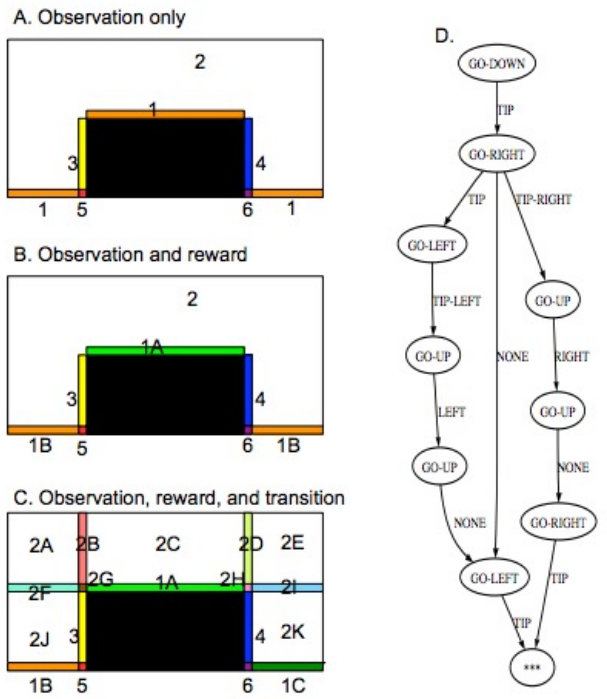


Fig. 1. A. Observation partition; B. Observation and reward partition; C. Closed partition; D. Partial policy graph for robot starting in an unknown state above the table, with a deterministic transition and observation model.

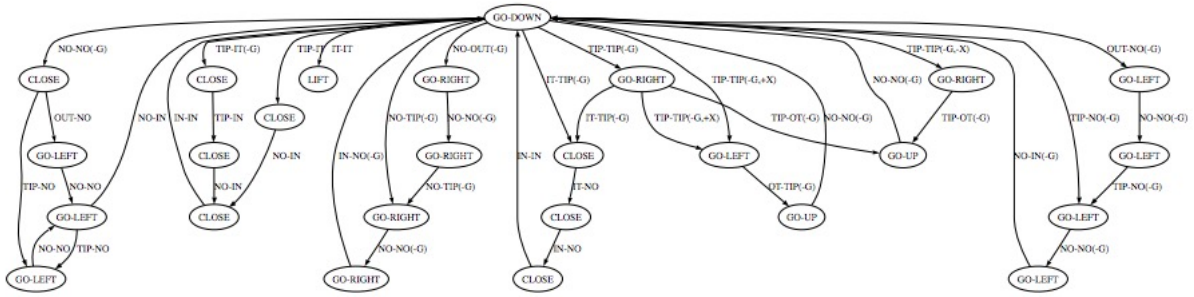


Fig. 5. Two-finger grasping policy for deterministic model.

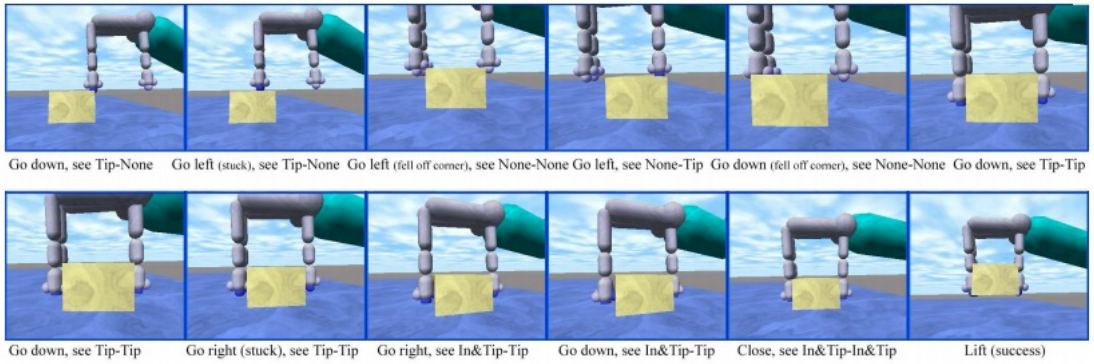


Fig. 6. Sample run of two-finger grasp policy in high-fidelity simulation.

https://people.csail.mit.edu/lpk/papers/pomdp-grasp-final.pdf

Belief Space MDPs

- Since we don't know the state, the set of future states and observations are *not* independent of past observations.
- The statistic over the reward (e.g., $E_{s_0:T}[\sum_t \gamma^t r(s_t)]$ cannot be computed without the entire history of actions and observations.
  - We can't compute $p(s_{0:t})$, can only compute $p(s_{0:t}|z_{0:t}, a_{0:T})$
  - Intractable to condition on the entire history of observations.
- Introduce a notion of a belief $b_t = p(s_{0:t}|z_{0:t}, a_{0:t})$ that expresses the agent's belief over the state derived from observations $z_{0:t}$ and actions $a_{0:t}$.
- The belief $b_t = p(s_{0:t}|z_{0:t}, a_{0:t})$ can be computed recursively, and serves as a sufficient statistic to determine future expected rewards.

MDP: Notation Recap

A Markov decision process can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where

- $\mathcal{S}$ is a finite set of states of the world;
- $\mathcal{A}$ is a finite set of actions;
- $T : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is the *state-transition function*, giving for each world state and agent action, a probability distribution over world states (we write $T(s, a, s')$ for the probability of ending in state s', given that the agent starts in state s and takes action a); and

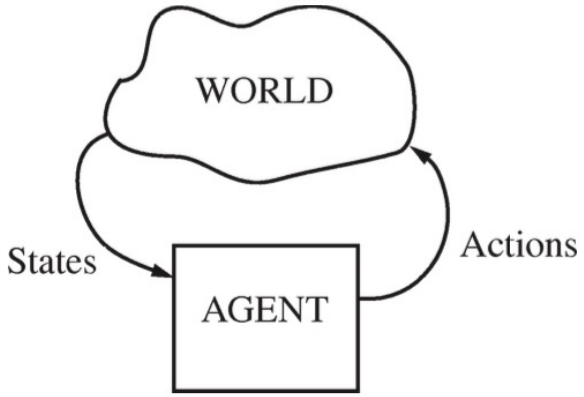


Fig. 1. An MDP models the synchronous interaction between agent and world.

POMDP Notation (from AIJ paper)

3.1. POMDP *framework*

A partially observable Markov decision process can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \Omega, O \rangle$, where

- $\mathcal{S}$, $\mathcal{A}$, T, and R describe a Markov decision process;
- Ω is a finite set of observations the agent can experience of its world; and
- $O : \mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$ is the *observation function*, which gives, for each action and resulting state, a probability distribution over possible observations (we write $O(s', a, o)$ for the probability of making observation o given that the agent took action a and landed in state s').

A POMDP is an MDP in which the agent is unable to observe the current state. Instead, it makes an observation based on the action and resulting state.[4] The agent's goal remains to maximize expected discounted future reward.

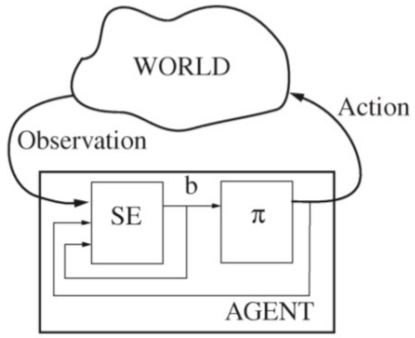


Fig. 2. A POMDP agent can be decomposed into a state estimator (SE) and a policy (π).

POMDP as a Belief State MDP

MDPs

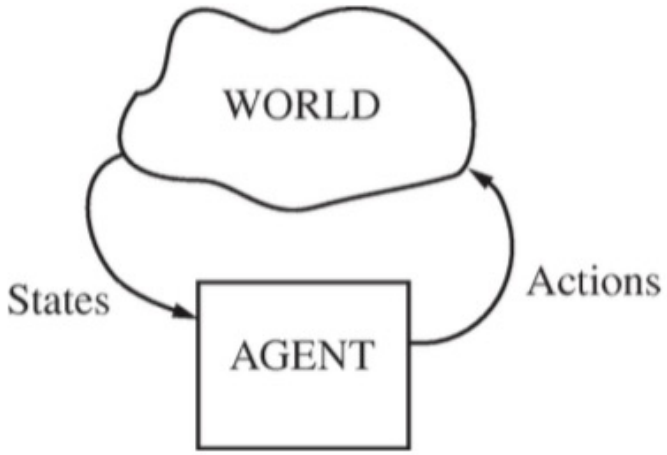


Fig. 1. An MDP models the synchronous interaction between agent and world.

POMDPs

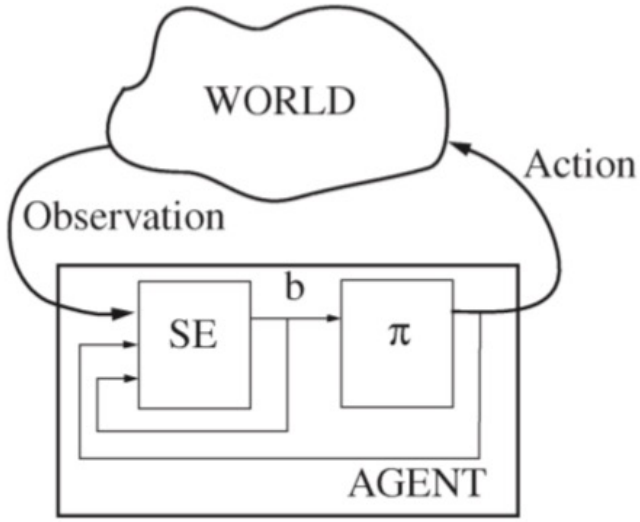


Fig. 2. A POMDP agent can be decomposed into a state estimator (SE) and a policy (π).

Belief State MDP

- Belief State Update
  - As actions are taken and observations received, we need to update our belief state
- State transitions
  - From one belief state to another, given an action
- Reward function
  - For a belief state, from original reward function on world states.
- Observation Model
  - In this derivation, the observation is emitted after the agent reaches the state s' after taking action a.

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

- $\tau(b, a, b')$ is the state-transition function, which is defined as

$$\tau(b, a, b') = \Pr(b' \mid a, b) = \sum_{o \in \Omega} \Pr(b' \mid a, b, o)\Pr(o \mid a, b),$$

where

$$\Pr(b' \mid b, a, o) = \begin{cases} 1 & \text{if } \mathrm{SE}(b, a, o) = b' \\ 0 & \text{otherwise;} \end{cases}$$

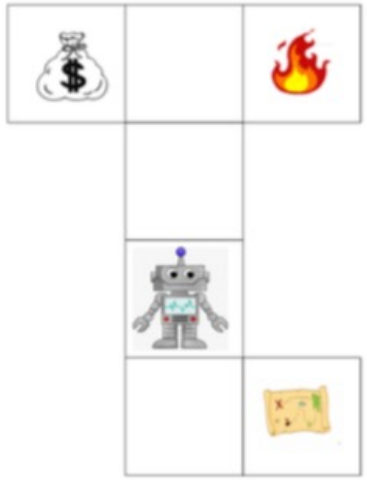
- $\rho(b, a)$ is the reward function on belief states, constructed from the original reward function on world states:

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

Illustrative POMDP Problems

- There are some famous POMDP benchmark problems that illustrate key ideas.
  - Tiger domain:

    - World is stochastic.
    - Agent doesn't know which door the tiger lurks behind.
    - Agent can either listens to the tiger roar (-1), and modify its estimate of which door the tiger is behind, or open a door and escape ($+10$) or get eaten (-100).
    - Problem forces the tradeoff between listening (gathering more information), or trusting the estimate to open a particular door.
  - Heaven & Hell:



    - World is deterministic.
    - Agent doesn't initially know which side of the T junction has $+100$ reward or -100 reward.
    - Map at the bottom of the L describes which side has which reward.
    - Problem forces active information gathering.

Tiger Example: Details

- There are two states: tiger-left and tiger-right.
- There are three actions: listen, open-left and open-right.
- The transition probabilities are as follows:
  - listen: the state self-transitions with probability 1. That is,

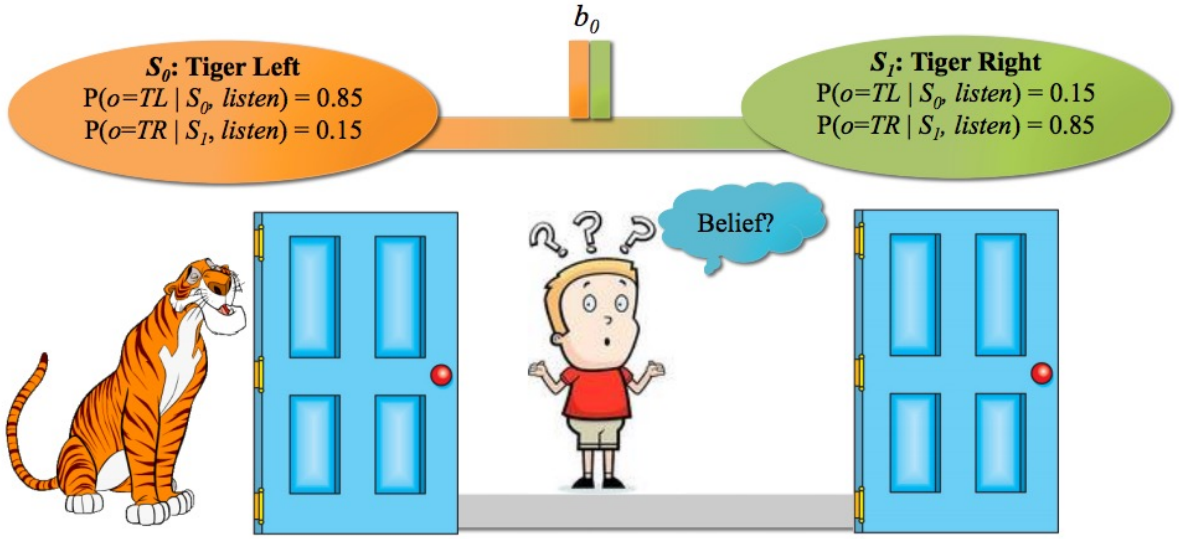
$$p(s_i|\text{listen}, s_j) = \begin{cases} 1 & \forall s_i == s_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

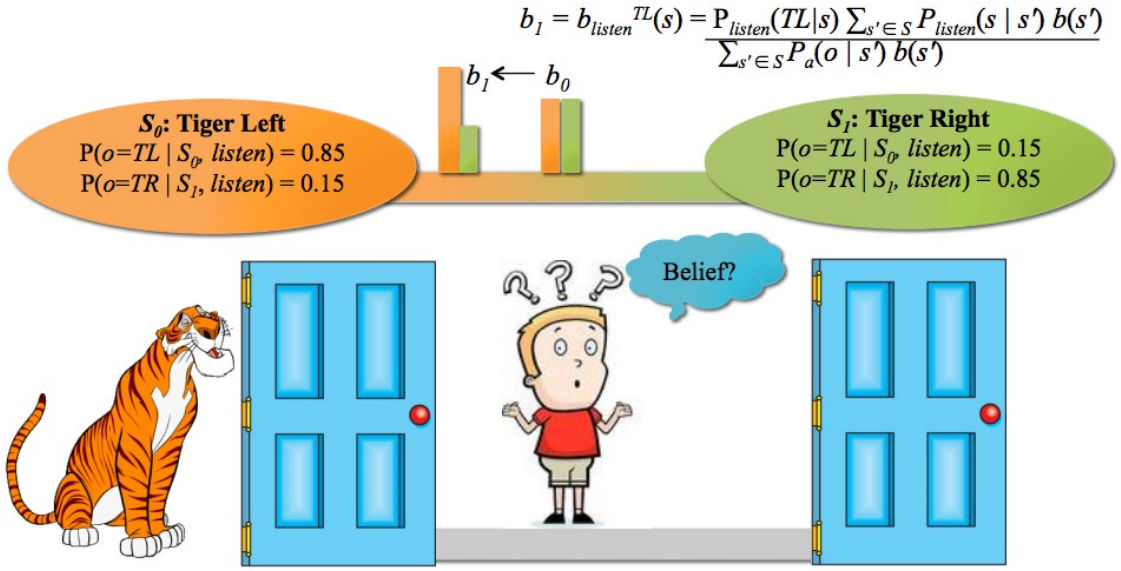
  - open-left, open-right : transition to either tiger-left or tiger-right with probability 0.5
- There are two observations: heard-left, heard-right
- The observation probabilities are as follows:
  - After the listen action:

$$p(\text{heard-left}|\text{listen}, \text{tiger-left}) = 0.85 \tag{2}$$
$$p(\text{heard-right}|\text{listen}, \text{tiger-left}) = 0.15 \tag{3}$$
$$p(\text{heard-left}|\text{listen}, \text{tiger-right}) = 0.15 \tag{4}$$
$$p(\text{heard-right}|\text{listen}, \text{tiger-right}) = 0.85 \tag{5}$$

  - After the open-left, open-right actions: heard-left and heard-right both have probability 0.5
- The reward function is as follows:
  - $R(s_i, \text{listen}) = -1$
  - $R(\text{tiger-left}, \text{open-right}) = 10$
  - $R(\text{tiger-right}, \text{open-right}) = -100$
  - $R(\text{tiger-left}, \text{open-left}) = -100$
  - $R(\text{tiger-right}, \text{open-left}) = 10$
- Our initial state distribution is $p(s_i) = 0.5\ \forall s_i$.
- Our discount factor is $\gamma = 0.75$.

Updating the Belief

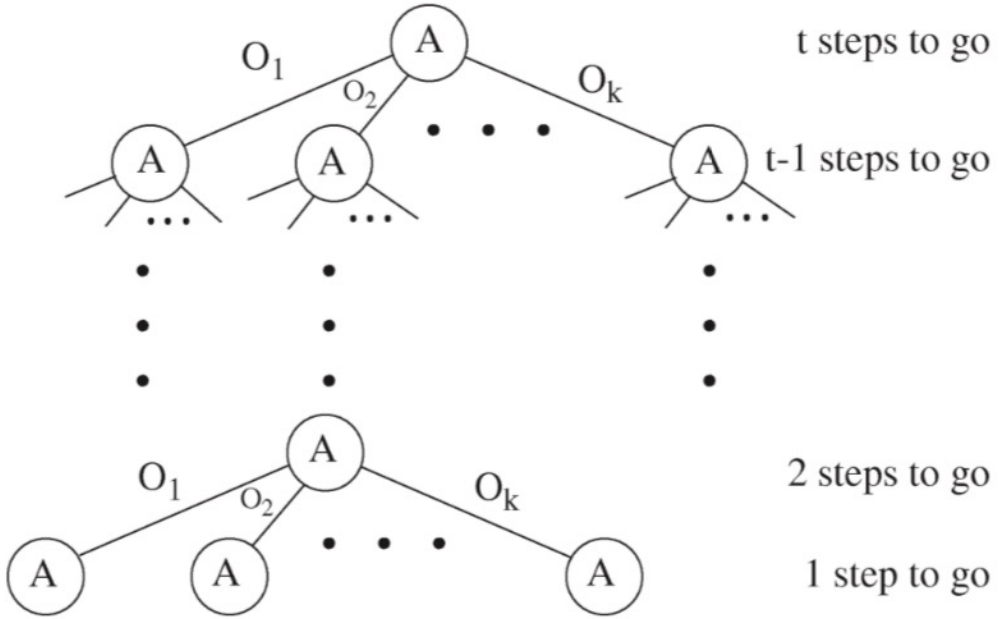




Updating the belief over the state using action and observation.
Action: Listen and Observation: Hear-Left

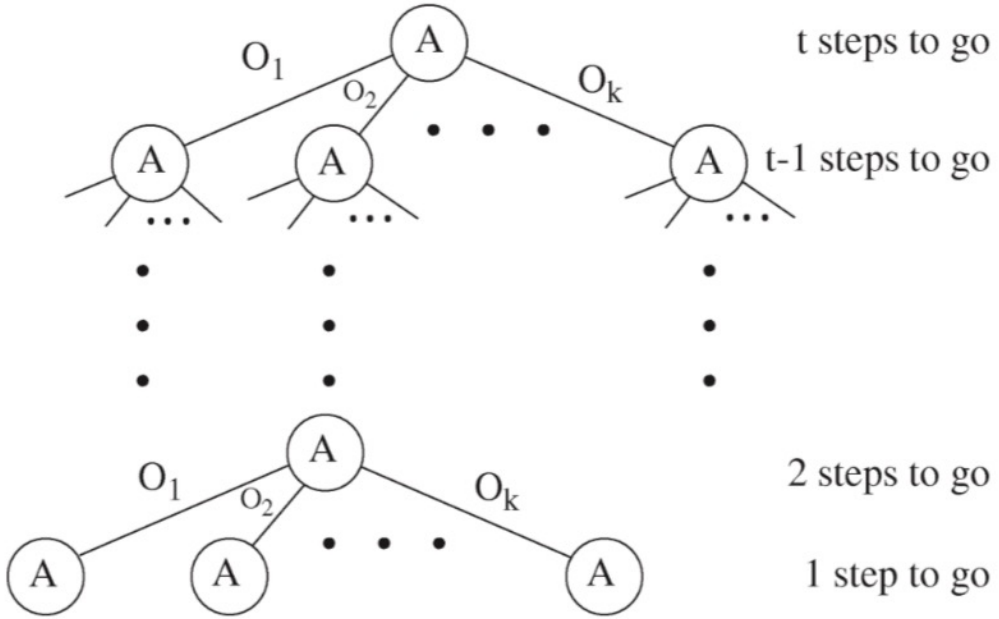
How to decide actions in a POMDP?

- Policy Trees
  - If the agent has only one step, then it can only take one action.
  - With two steps to go, it can take an action, receive an observation and take another action (depending on the previous observation)
  - Can extend this to t-steps.
- Root of the Policy Tree
  - The root of the tree a(p) is the action we are considering.
  - Finally, we will execute that action.
  - There can be many options for actions that we can select from.
- How do we select an action?
  - Determine the value that one can obtain by taking an action.
  - Consider the future of observations and actions that we can obtain.



How to decide actions in a POMDP?

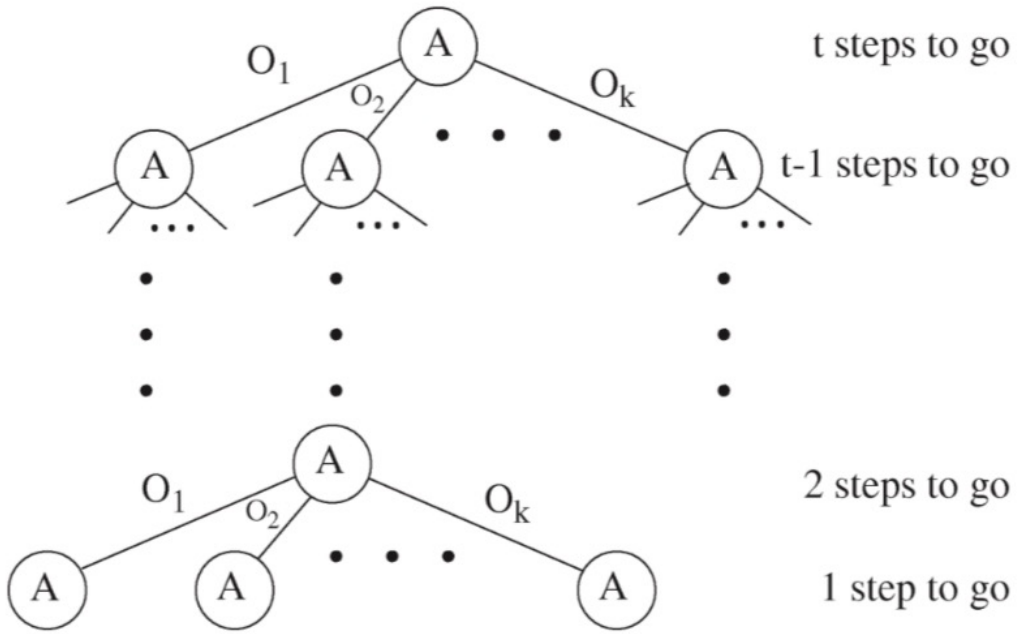
- **Interpret as a "conditional" plan**
  - The agent knows what to do for any observation that can arrive.
- **Essentially equivalent to "decision trees**
  - As used in decision theory to represent a sequential decision policy; but not to "decision trees" as used in machine learning to compactly represent a single-stage decision rule.



For MDP, we could compute the optimal value function and then used to determine the optimal policy.
How to do the same for POMDPs?
Determine the value function and then determine the policy.

Value Function for POMDPs

What is the expected discounted value to be gained from executing a policy tree p?



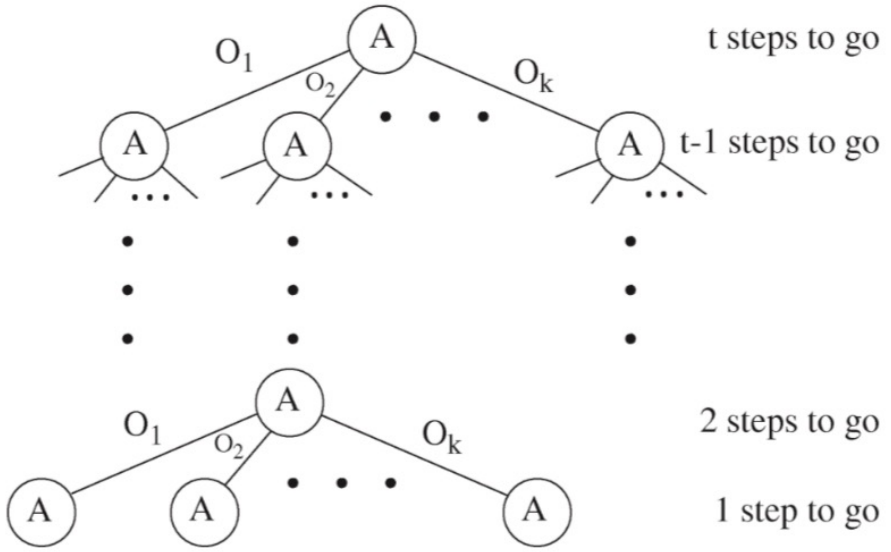
1-Step Case

- If p is a 1-step policy tree (a single action). The value of executing that action in state s is below.
- Here, a(p) is the action specified at the top node of the policy tree, p.

$$V_p(s) = R\big(s, a(p)\big)$$

Value Function for POMDPs

What is the expected discounted value to be gained from executing a policy tree p?



T-step Case

- More generally, if p is a t-step policy tree then:

$$
\begin{aligned}
V_p(s) &= R\big(s, a(p)\big) + \gamma \cdot \big(\text{Expected value of the future}\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} \Pr\big(s' \mid s, a(p)\big) \sum_{o_i \in \Omega} \Pr\big(o_i \mid s', a(p)\big) V_{o_i(p)}\big(s'\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} T\big(s, a(p), s'\big) \sum_{o_i \in \Omega} O\big(s', a(p), o_i\big) V_{o_i(p)}\big(s'\big)
\end{aligned}
$$

Detailed Description

Now, what is the expected discounted value to be gained from executing a policy tree p? It depends on the true state of the world when the agent starts. In the simplest case, p is a 1-step policy tree (a single action). The value of executing that action in state s is

$$V_p(s) = R\big(s, a(p)\big)$$

where $a(p)$ is the action specified in the top node of policy tree p. More generally, if p is a t-step policy tree, then

$$\begin{aligned} V_p(s) &= R\big(s, a(p)\big) + \gamma \cdot \big(\text{Expected value of the future}\big) \\ &= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} \Pr\big(s' \mid s, a(p)\big) \sum_{o_i \in \Omega} \Pr\big(o_i \mid s', a(p)\big) V_{o_i(p)}\big(s'\big) \\ &= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} T\big(s, a(p), s'\big) \sum_{o_i \in \Omega} O\big(s', a(p), o_i\big) V_{o_i(p)}\big(s'\big) \end{aligned}$$

where $o_i(p)$ is the $(t-1)$-step policy subtree associated with observation o_i at the top level of a t-step policy tree p. The expected value of the future is computed by first taking an expectation over possible next states, s', then considering the value of each of those states. The value depends on which policy subtree will be executed which, itself, depends on which observation is made. So, we take another expectation, with respect to the possible observations, of the value of executing the associated subtree, $o_i(p)$, starting in state s'.

Value Functions and Alpha Vectors

- Value function for the finite-horizon POMDP will be the **supremum** of the value functions of each policy tree.
  - The value of each tree is a hyperplane.
  - The value function can therefore be represented by a set of hyper-planes defined by the policy trees.
- Alpha-vectors
  - The hyper-planes are called the alpha vectors
  - Nomenclature: alpha-vectors is often used to denote both the coefficients of the value hyper-planes and the value hyperplanes themselves.

Value Functions and Alpha Vectors

- Value of executing a policy tree p from some belief state b.
  - Expectation over world states of executing p in each state.
  - The value is a function of the belief.

$$V_p(b) = \sum_{s \in \mathcal{S}} b(s) V_p(s)$$

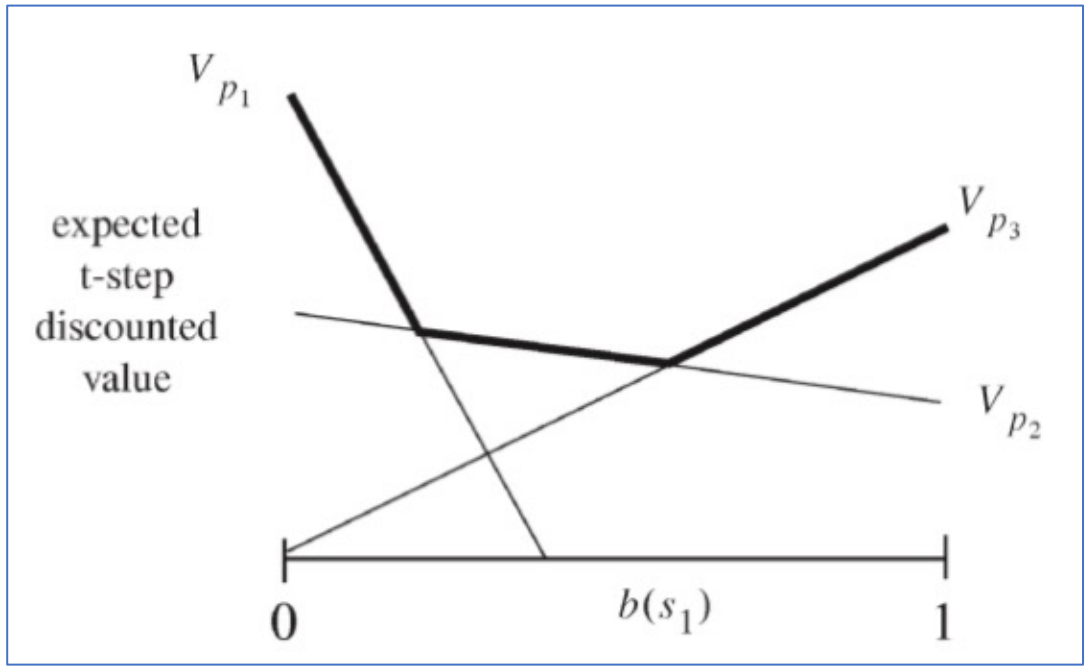
- Alpha-vector
  - Note sometimes alpha is also used to denote the value hyperplanes.

$$\alpha_p = \langle V_p(s_1), \ldots, V_p(s_n) \rangle \quad \text{then } V_p(b) = b \cdot \alpha_p$$

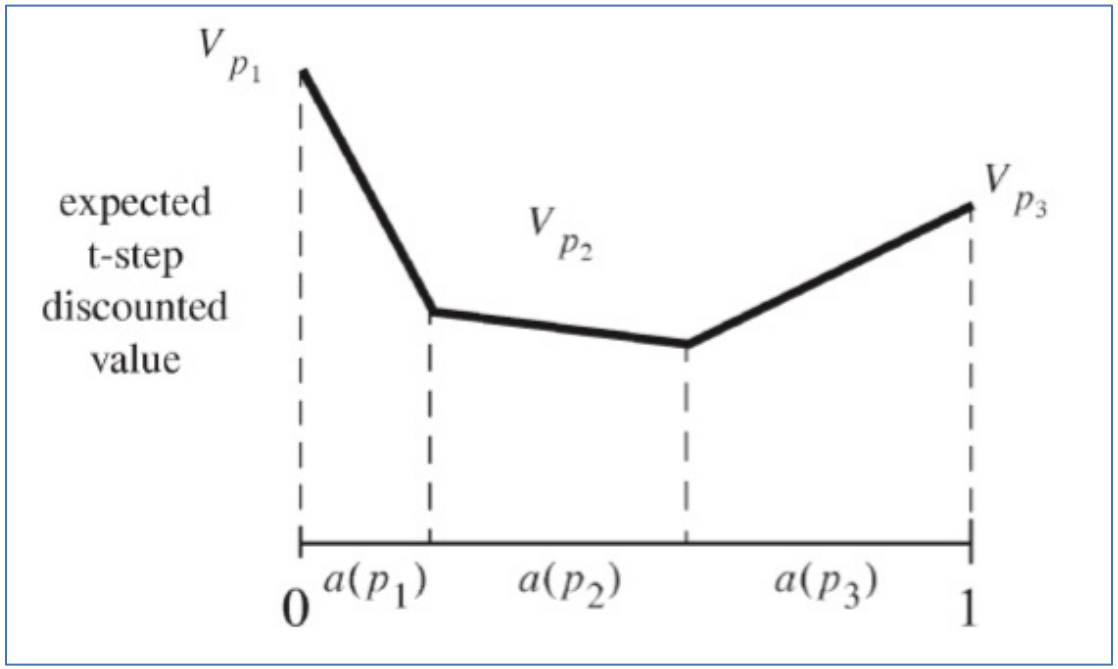
- To construct the optimal t-step policy,
  - Necessary to execute different policy trees from different initial belief states.
  - Finally, we assign the root of the tree whole policy given the best return at a particular belief.
  - Note: need to do max taking into account the belief distribution.

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

Value Functions and Policy Trees



Optimal t-step value function is the upper surface of the value functions associated with all t-step policy trees. Arises due to the max while computing the value.



Optimal t-step situation-action mapping. In which region of the belief space, which policy tree is applicable. Take the root of the tree whose tree gives the highest return for that belief.

Note: We will only execute the first action a(p) but we evaluated many $a(p_1)$, $a(p_2)$, … etc.
Further, we account for the uncertainty in state. Hence, given the "belief" we prescribe an action.

Optimal Policy and Policy Trees

- Optimal policy
  - Different policy trees can be associated with different regions of the belief space.
  - The optimal policy consists of a division of belief space into convex regions each assigned its own policy tree.

- Example
  - When there are three world states, the belief state is determined by two values.
  - Because of simplex constraints summing to 1). The distribution adds to 1.
  - The belief space is a triangle in the 2-space. The value function can be assigned to each point in the Z direction.

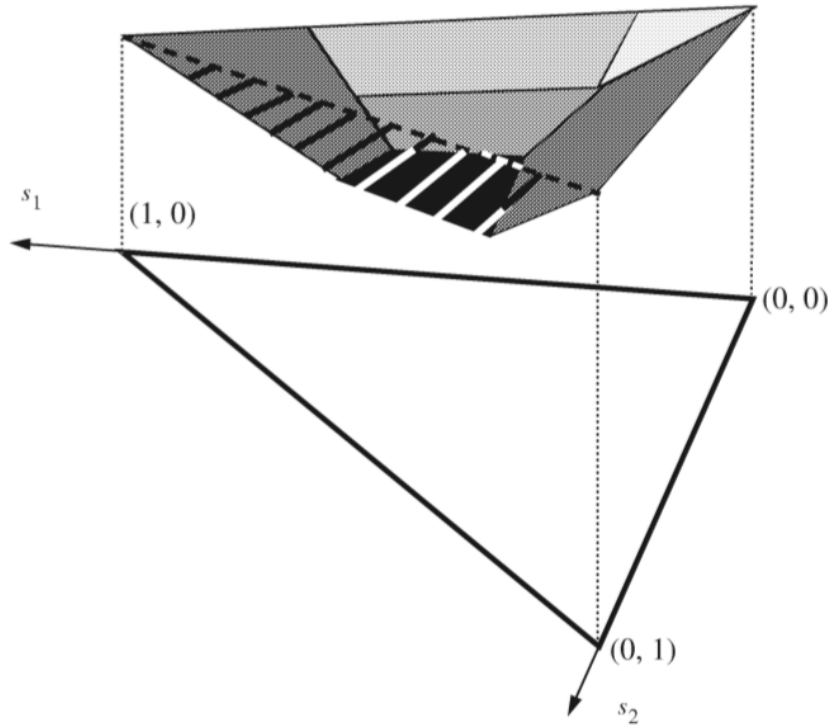
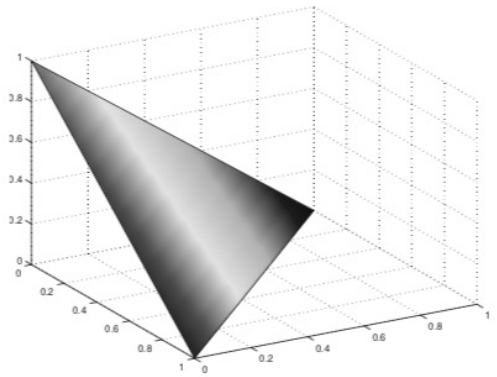


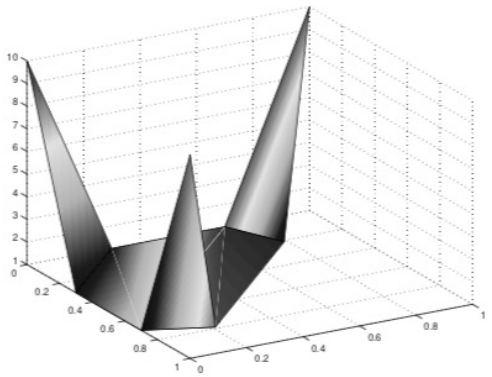
Fig. 6. A value function in three dimensions is made up of the upper surface of a set of planes.

Policies

- The policy tree is **not** the complete policy
  - There are three such policy trees that describe the full one-step policy.
  - The horizontal axis of the graph is the belief b, and the vertical axis is the expected value for each belief.



(a) A belief space (distribution adds to one)



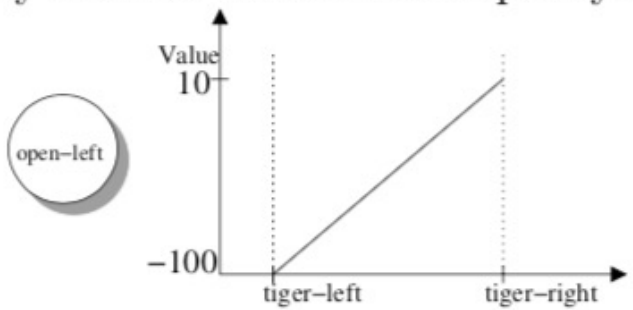
(b) A value function defined over the belief space.

When there are three world states, the belief state is determined by two values (because of simplex constraints summing to 1). The belief space is a triangle in the 2-space. The value function can be assigned to each point in the Z direction.

- Can associate different policy trees with different regions of belief space. The optimal policy consists of a division of belief space into convex regions each with its own policy tree.
  - In general, a belief space of $|\mathcal{S}|$ states has $|\mathcal{S}| - 1$ dimensions, as does the value function
  - For the purposes of visualization, the set of beliefs that constitutes the belief space shown in (a) has been projected onto the XY plane in (b). The value function then rises along the positive Z axis.
- Each point in the belief space corresponds to a specific distribution, and the value function at that point gives the expected reward of the policy starting from this belief.

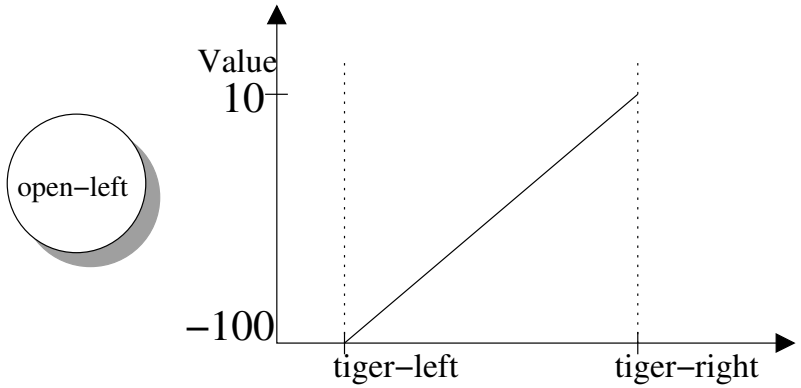
The One-Step Policy

- Consider the simplest task of selecting one-step policy or making the choice of the first action.
- If we have a choice of actions, we should select the one with the higher value function.
- The value function if the expected value of the immediate reward for the one-step case.
- POMDPs use a notion of policy trees to understand a policy.



- The value function for this policy tree is the expected value of the immediate reward function under this policy tree for action *oil*, where the immediate reward for this action (specified in the model) is
  $R(\text{tiger-left}, \text{open-left}) = -100, R(\text{tiger-right}, \text{open-left}) = 10.$
- The value function for the one-step policy tree is a single line $\Rightarrow$ the value function is the *expected* immediate reward and expectation is linear.
- Can depict the belief space for a 2 state problem using a single axis if we parameterise the belief b as the pair $(p(\text{tiger-right}), 1 - p(\text{tiger-right}))$.

Linear Interpolation In Belief Space



- Another way to see the linearity is to remember that the value function maps beliefs to rewards:

$$V : B \times A \mapsto \mathcal{R} \tag{6}$$

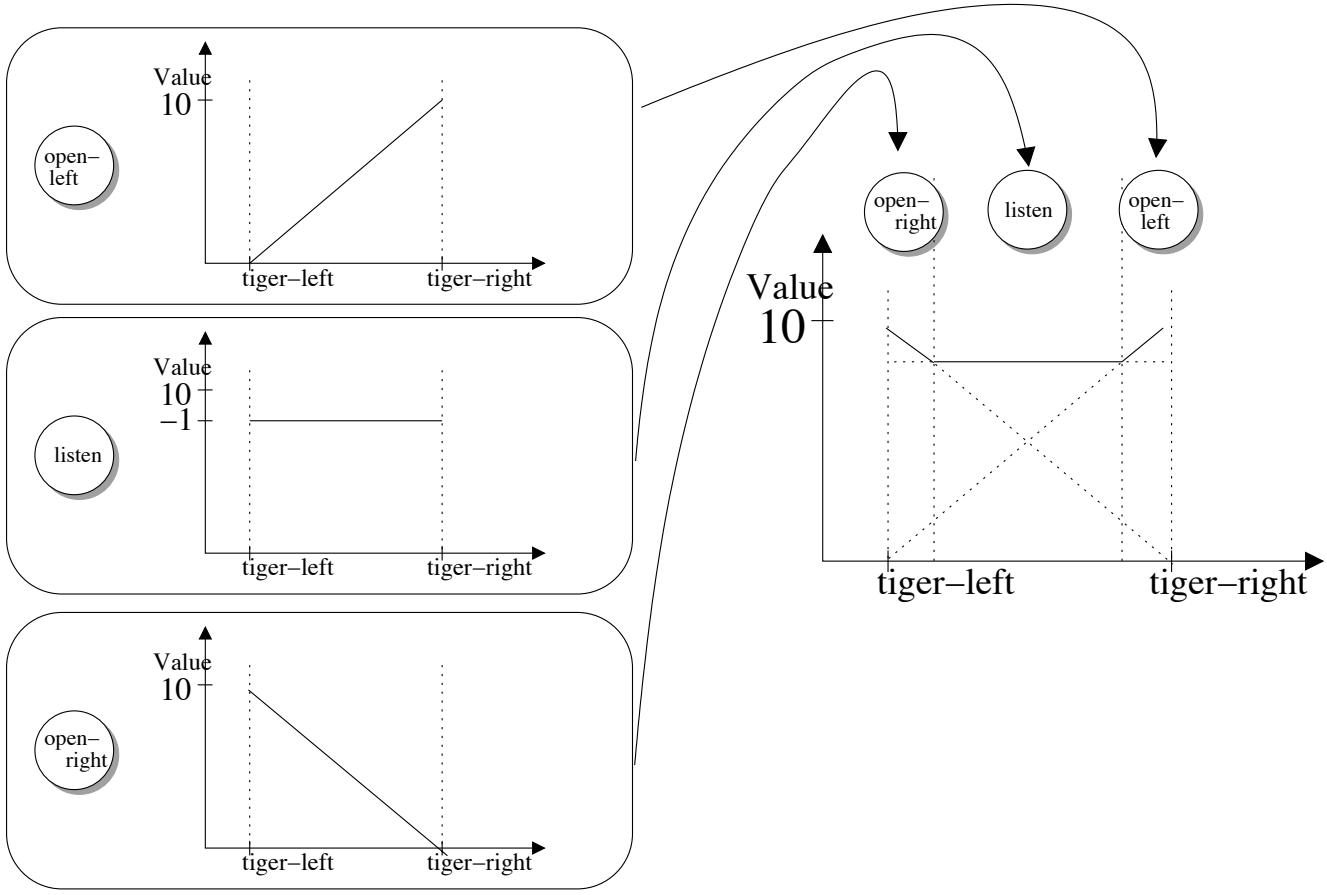
and is defined as

$$V(b) = \sum_{S} p(s)V(s), \tag{7}$$

- Means that to specify the value function for any one-step policy, we need only specify the value at the states
- The complete value function for that policy tree will be the linear interpolation across the belief space.

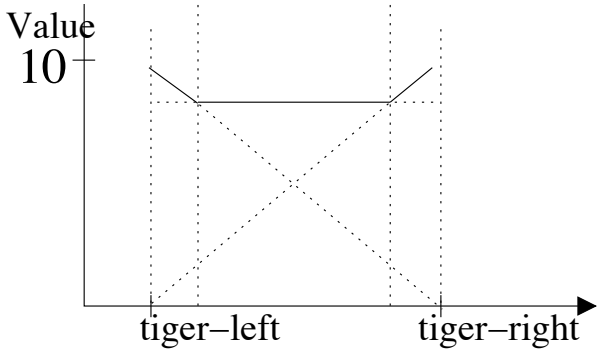
The Exact One-Step Policy

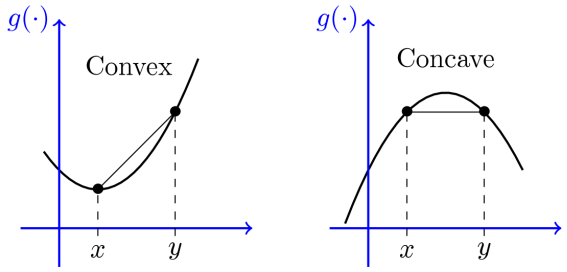
- The complete one step policy
  - Is generated by determining which one-step policy tree is optimal for regions of the belief space.
  - The action at the root describes the action to take for each belief in that region of the belief space.
- The complete value function
  - Is the supremum of the value functions for the trees and determines which policy tree is assigned to which region of belief space.
- Example
  - In the middle there is greater uncertainty and listening is preferred.
  - At the edges where there is high uncertainty, opening is preferred.



Convexity

- If the value function is composed of the supremum of hyperplanes, then the value function is piece-wise linear.
  - ⇒ The value function must be convex.
    - That the supremum of a set of vectors is convex follows from the definition of convexity
- Also makes sense intuitively: in the middle of the belief space, where we are less sure about where the tiger is, the listen action has the highest expected value.
- At the edges of the belief space, we are more confident about where the tiger is and can go ahead and open the door.
  - Would expect the edges of the belief space to be higher value in general
  - We have less need for the costs of information gathering, and less likely to make a mistake and choose a high-cost action because we didn't know the true state.
- Since the α-vectors are linear, and the value function is the supremum of the α-vectors, we can represent the entire value function by evaluating policy trees only for beliefs where the state is known perfectly (i.e., at the corners of the belief simplex).

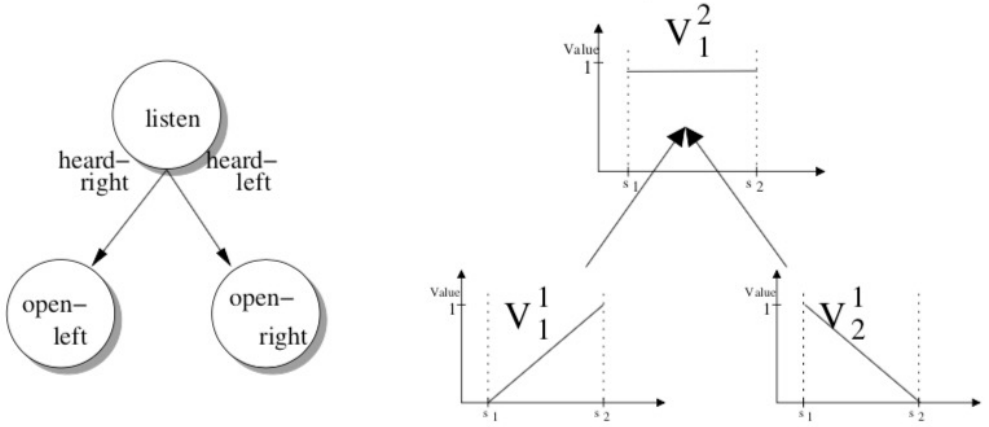




The value function (max over b.alpha_p) for the POMDP is convex.

Two Step Planning

- Having solved the one-step planning problem, we can use this solution to solve the two-step planning problem.
- First see how to develop a two-step policy tree and then see how to compute the entire two-step policy.
- Just as in the one-step case, entire two-step policy will be given by:
  1. computing all possible two-step trees
  2. assigning each tree to the region of belief space where its associated α-vector dominates
  - Once the two-step policy is created, we can dispense with the trees and α-vectors from the one-step policy.
  - The $t-1$ policy trees are only useful for computing the t policy.



- Figure depicts a horizon 2 policy tree on the left.
  - The policy tree consists of an initial action, an (unpredictable) observation, and a subsequent action.
- Can compute the value of this policy tree from the value functions of the one-step policy trees.

Example: Value of executing a Policy Tree

- What is the expected discounted value to be gained from executing a policy tree p?
  - This is also called "backing up" of alpha-vectors.
  - The term "backing up" comes from the equivalence to Bellman backups studied earlier.
  - In the example, we compute the value for action *listen* for the states *tiger-left* and *tiger-right.*
- How to backup alpha vectors?
  - The immediate reward of the root action at each state R(s, listen)
  - And the value of each subtree V^i_1,
    - weighted by the probability of the observation zi
    - and the probability of being at each state given the observation.

$$\alpha(\text{tiger-left}) = R(\text{tiger-left}, \text{listen}) + \gamma\Big(p(\text{tiger-left}|\text{tiger-left}, \text{listen})p(\text{heard-right}|\text{tiger-left}, \text{listen})V_1^{\text{open-left}}(\text{tiger-left}) + p(\text{tiger-right}|\text{tiger-left}, \text{listen})p(\text{heard-right}|\text{tiger-right}, \text{listen})V_1^{\text{open-left}}(\text{tiger-right}) + p(\text{tiger-left}|\text{tiger-left}, \text{listen})p(\text{heard-left}|\text{tiger-left}, \text{listen})V_1^{\text{open-right}}(\text{tiger-left}) + p(\text{tiger-left}|\text{tiger-left}, \text{listen})p(\text{heard-left}|\text{tiger-right}, \text{listen})V_1^{\text{open-right}}(\text{tiger-right})\Big)$$

$$\Rightarrow \alpha(\text{tiger-left}) = -1 + 0.75\Big(1 \times 0.15 \times -100 + 0 \times 0.85 \times 10 + 1 \times 0.85 \times 10 + 0 \times 0.15 \times -100\Big)$$

$$\Rightarrow \alpha(\text{tiger-left}) = -5.875$$

$$\alpha(\text{tiger-right}) = -1 + 0.75\Big(0 \times 0.15 \times -100 + 1 \times 0.85 \times 10 + 0 \times 0.85 \times 10 + 1 \times 0.15 \times -100\Big)$$

$$\Rightarrow \alpha(\text{tiger-right}) = -5.875$$

Note: one way to think of the backups is that we are estimating the values at the extremes of the belief surface. Once we know at the extremes, the remaining can be interpolated due to linearity. The above is for a particular policy tree. Finally, we compute the maxima.

Two-step Policy from Policy Trees

- While performing the recursion, need to compute all possible two-step trees
- And then assign each tree to the region of belief space where it has superiority.

Computing the value of a policy tree requires examining all the possible sub-trees.

$$
\begin{aligned}
V_p(s) &= R\big(s, a(p)\big) + \gamma \cdot \big(\text{Expected value of the future}\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} \Pr\big(s' \mid s, a(p)\big) \sum_{o_i \in \Omega} \Pr\big(o_i \mid s', a(p)\big) V_{o_i(p)}\big(s'\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} T\big(s, a(p), s'\big) \sum_{o_i \in \Omega} O\big(s', a(p), o_i\big) V_{o_i(p)}\big(s'\big)
\end{aligned}
$$

Figure: The set of 9 possible subtrees of this two-step policy tree.

POMDP Value Iteration in Practice

- It is infeasible to enumerate every possible t-step plan to find the one that maximizes the value function.
- In practice, we iterate over all the one-step plans and toss out the plans that are not optimal for any initial belief state (called pruning).
- The process repeats till the planning horizon is reached.
- The alpha-vectors provide a compact representation of the policy.
- POMDP solvers clearly manage the alpha vectors. They prune away less promising vectors, compress, estimate bounds etc.

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

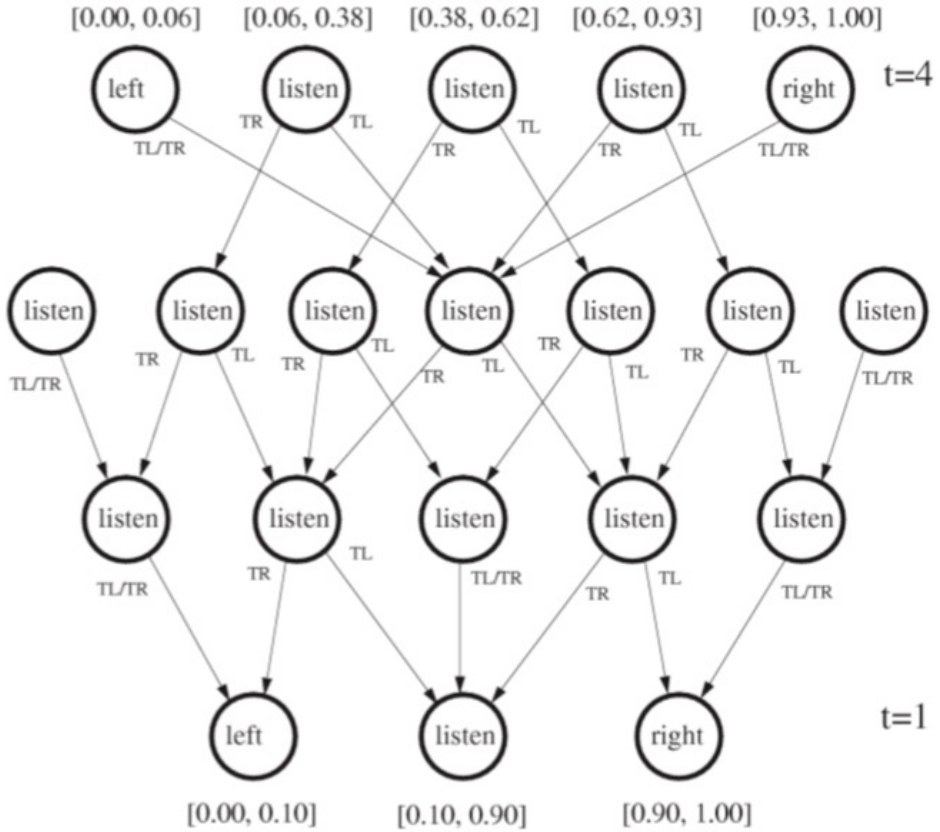
Policies for the Tiger Problem

Optimal policies for one time step

- Prefer to listen when you are highly uncertain.
- Prefer to open the respective door if the belief is highly certain.

Optimal policies for four time steps

- The agent will choose to open a door for some belief states.



Tractable VI in POMDPs - Pruning

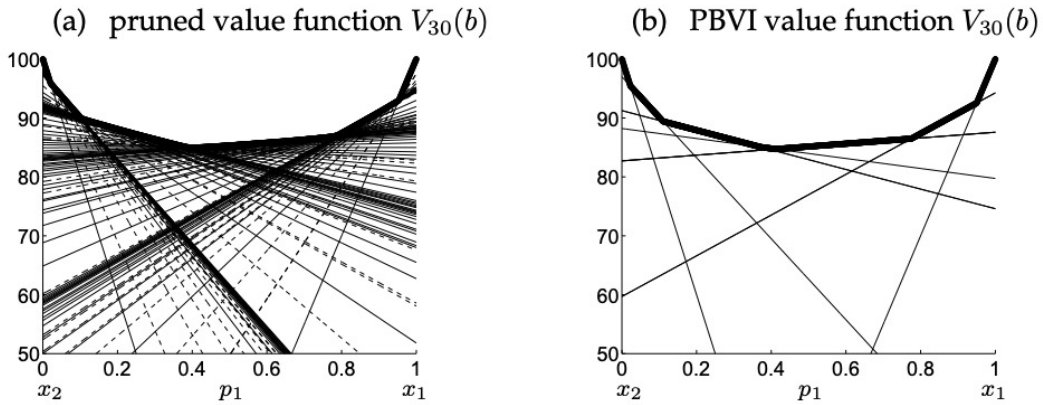


Figure 15.7 The benefit of point-based value iteration over general value iteration: Shown in (a) is the exact value function at horizon $T = 30$ for a different example, which consists of 120 constraints, after pruning. On the right is the result of the PBVI algorithm retaining only 11 linear functions. Both functions yield virtually indistinguishable results when applied to control.

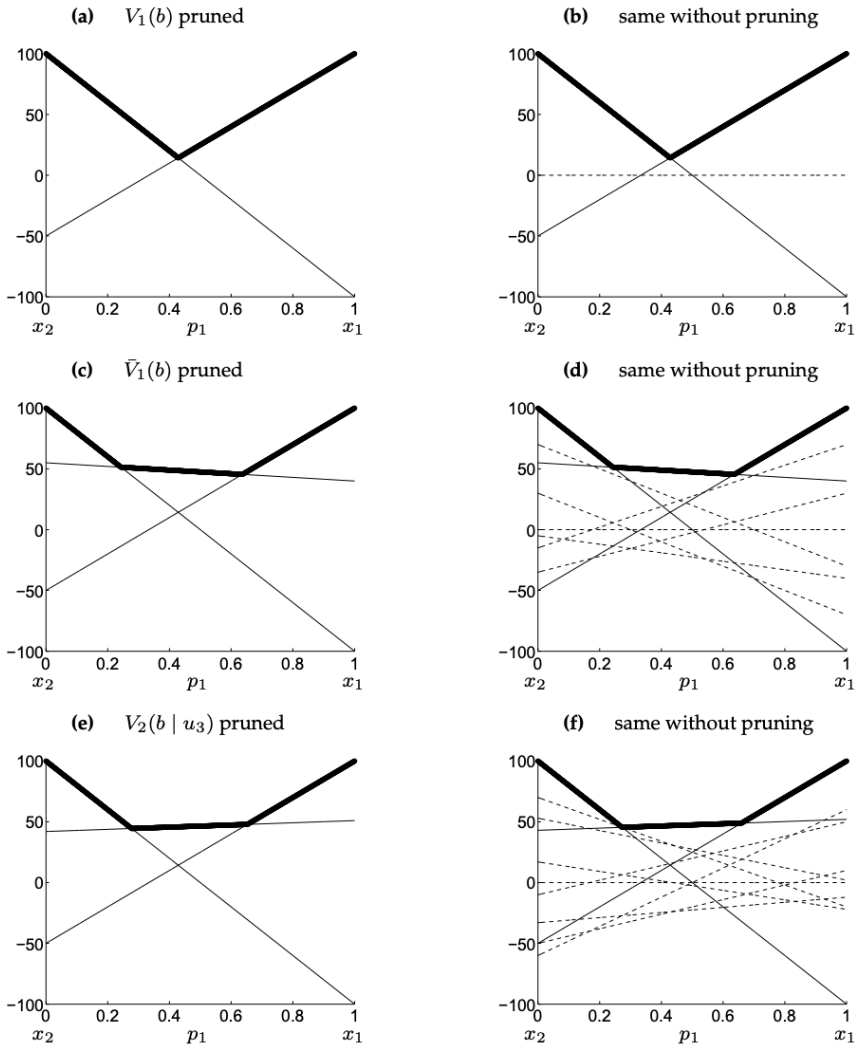


Figure 15.6 Comparison of an exact pruning algorithm (left row) versus a non-pruning POMDP algorithm (right row), for the first few steps of the POMDP planning algorithm. Obviously, the number of linear constraints increases dramatically without pruning. At $T = 20$, the unpruned value function is defined over $10^{547,864}$ linear functions, whereas the pruned one only uses 13 such functions.

Maintaining Beliefs via Particles

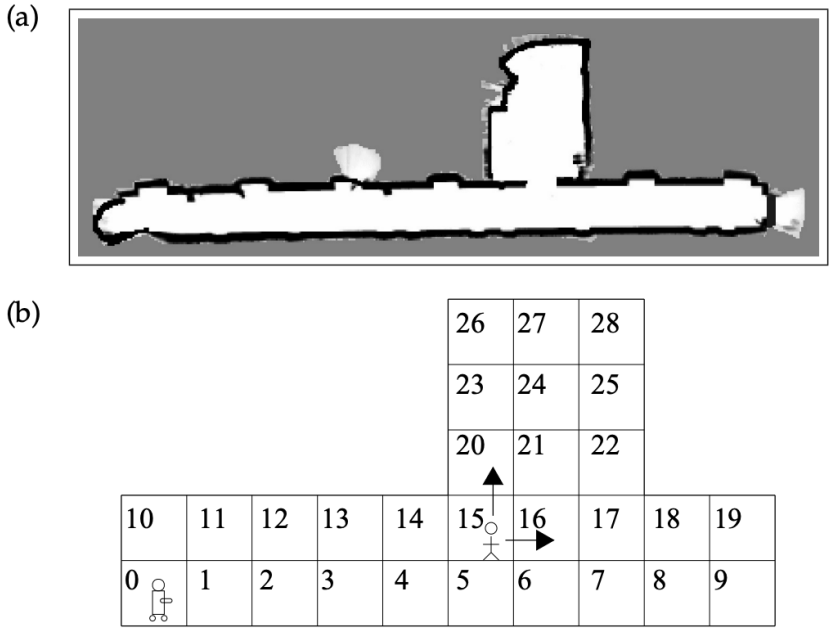


Figure 15.8 Indoor environment, in which we seek a control policy for finding a moving intruder. (a) Occupancy grid map, and (b) discrete state set used by the POMDP. The robot tracks its own pose sufficiently well that the pose uncertainty can be ignored. The remaining uncertainty pertains to the location of the person. Courtesy of Joelle Pineau, McGill University.

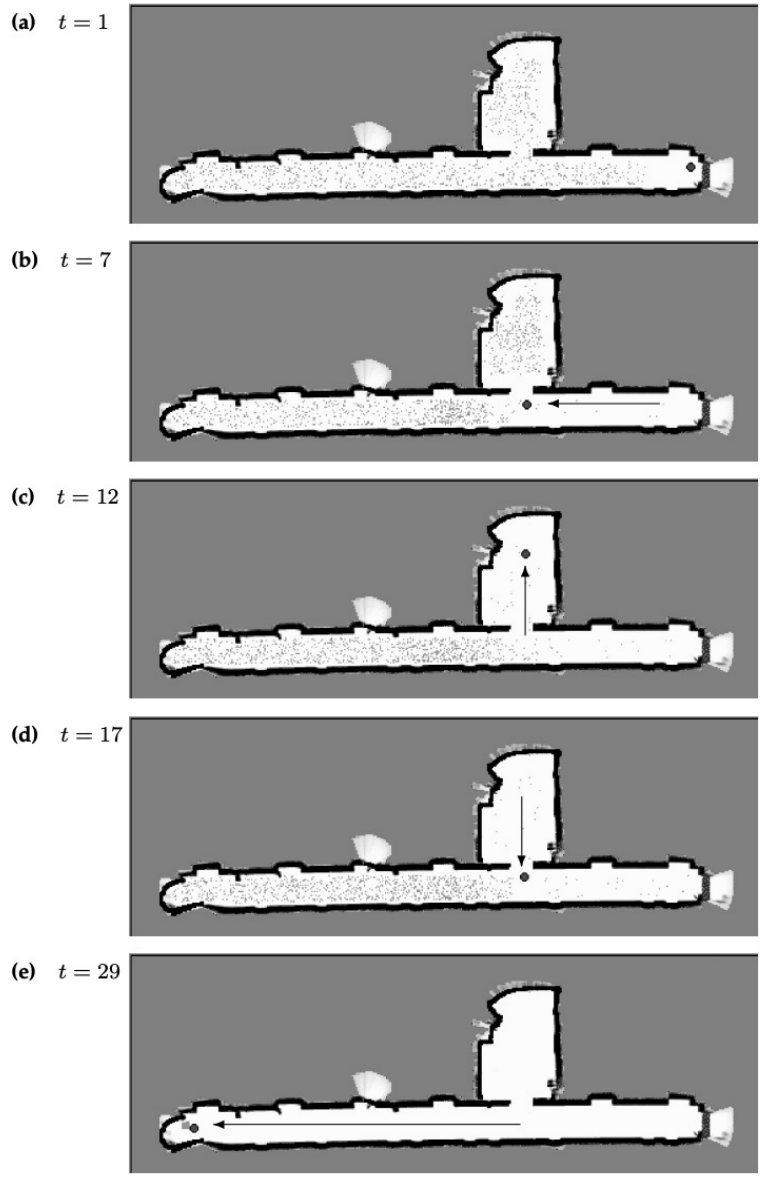


Figure 15.9 A successful search policy. Here the tracking of the intruder is implemented via a particle filter, which is then projected into a histogram representation suitable for the POMDP. The robot first clears the room on the top, then proceeds down the hallway. Courtesy of Joelle Pineau, McGill University.

Also called Monte Carlo POMDPs

Greedy Approaches

- Solve Underlying MDP
  - π_{MDP}: S → A; Q_{MDP}: S x A → $\Re$
- Choose Action Based on Current Belief State
  - **"most likely"** π_{MDP} (argmax$_s$(b(s))
  - **"voting"** argmax$_a$($\Sigma_{s \in S}$ b(s)δ(a, π_{MDP} (s))) where
    δ(a, b) = (1 if a=b; 0 otherwise)
  - **"Q-MDP"** argmax$_a$($\Sigma_{s \in S}$ b(s) Q_{MDP}(s, a))
- Essentially, try to act optimally as if the POMDP were to become observable after the next action

https://www.cs.cmu.edu/~ggordon/780-fall07/lectures/POMDP_lecture.pdf

Applications – Collecting Informative Samples

Information-Guided Robotic Maximum Seek-and-Sample in Partially Observable Continuous Environments

Genevieve Flaspohler,*[1,2] Victoria Preston,*[1,2] Anna P.M. Michel,[2] Yogesh Girdhar,[2] and Nicholas Roy[1]

***Abstract*— We present PLUMES, a planner for localizing and collecting samples at the global maximum of an *a priori* unknown and partially observable continuous environment. This "maximum seek-and-sample" (MSS) problem is pervasive in the environmental and earth sciences. Experts want to collect scientifically valuable samples at an environmental maximum (e.g., an oil-spill source), but do not have prior knowledge about the phenomenon's distribution. We formulate the MSS problem as a partially-observable Markov decision process (POMDP) with continuous state and observation spaces, and a sparse reward signal. To solve the MSS POMDP, PLUMES uses an information-theoretic reward heuristic with continuous-observation Monte Carlo Tree Search to efficiently localize and sample from the global maximum. In simulation and field experiments, PLUMES collects more scientifically valuable samples than state-of-the-art planners in a diverse set of environments, with various platforms, sensors, and challenging real-world conditions.**

I. Introduction

In many environmental and earth science applications, experts want to collect scientifically valuable samples of a

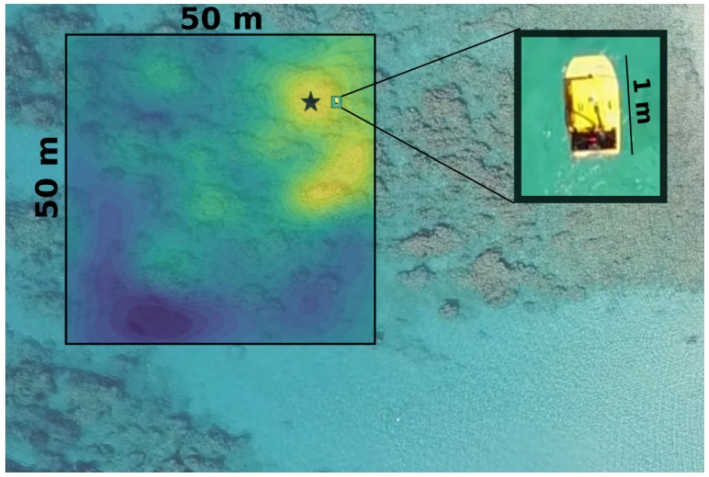


Fig. 1. **Coral head localization with an autonomous surface vehicle (ASV):** The objective of the ASV is to find and sample at the most exposed (shallowest) coral head in a region of Bellairs Fringing Reef, Barbados. Overlaid on the aerial photo is the *a priori* unknown bathymetry of the region (yellow is shallow, blue is deep). Equipped with an acoustic point altimeter, the ASV must explore to infer the location of the maximum (marked with a star) and then sample at that coral colony.

https://arxiv.org/pdf/1909.12216.pdf

Applications – Object Search

2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Multi-Object Search using Object-Oriented POMDPs

Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Lawson L.S. Wong, and Stefanie Tellex

***Abstract*— A core capability of robots is to reason about multiple objects under uncertainty. Partially Observable Markov Decision Processes (POMDPs) provide a means of reasoning under uncertainty for sequential decision making, but are computationally intractable in large domains. In this paper, we propose Object-Oriented POMDPs (OO-POMDPs), which represent the state and observation spaces in terms of classes and objects. The structure afforded by OO-POMDPs support a factorization of the agent's belief into independent object distributions, which enables the size of the belief to scale linearly versus exponentially in the number of objects. We formulate a novel Multi-Object Search (MOS) task as an OO-POMDP for mobile robotics domains in which the agent must find the locations of multiple objects. Our solution exploits the structure of OO-POMDPs by featuring human language to selectively update the belief at task onset. Using this structure, we develop a new algorithm for efficiently solving OO-POMDPs: Object-Oriented Partially Observable Monte-Carlo Planning (OO-POMCP). We show that OO-POMCP with grounded language commands is sufficient for solving challenging MOS tasks both in simulation and on a physical mobile robot.**

I. Introduction

A core capability of robots is to reason about multiple objects under uncertainty. A rescue robot, for example, may

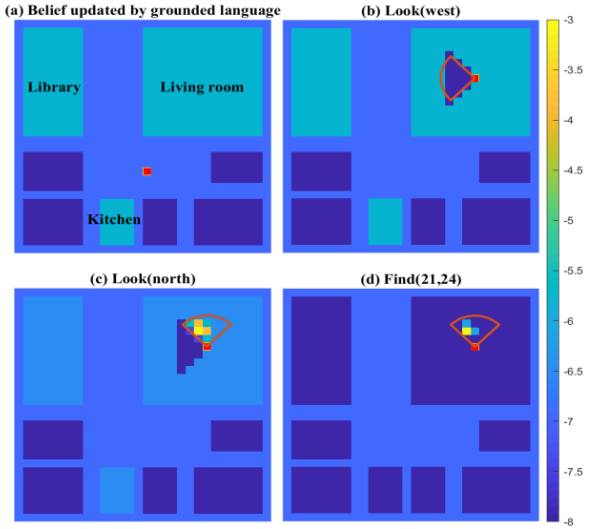


Fig. 1: The agent (red dot) must find the locations of one or more objects (yellow). (a) The belief after a language observation "Find the mugs in the library, living room, or kitchen." (b) and (c) show the belief after a sensor observations via `Look` actions. (d) The agent's belief converges to the true object location after a sequence of actions. (Probabilities in log scale.)

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8793888 and https://h2r.github.io/pomdp-py/html/examples.mos.html

Takeaways

- POMDPs consider the partial-observability of the state
  - In an MDP state is assumed to be known
  - In a POMDP, the agent receives noisy observations
- The notion of a belief state is central to POMDPs
  - A belief state is computed from past observations and actions.
  - POMDP is an MDP in the belief space.
- The complete value function for a policy tree is a linear interpolation across the belief space.
- The value function for a finite-horizon POMDP will be the supremum of the value functions for each policy tree.