COL864: Special Topics in AI
Semester II, 2020-21

POMDPs

Rohan Paul

Outline

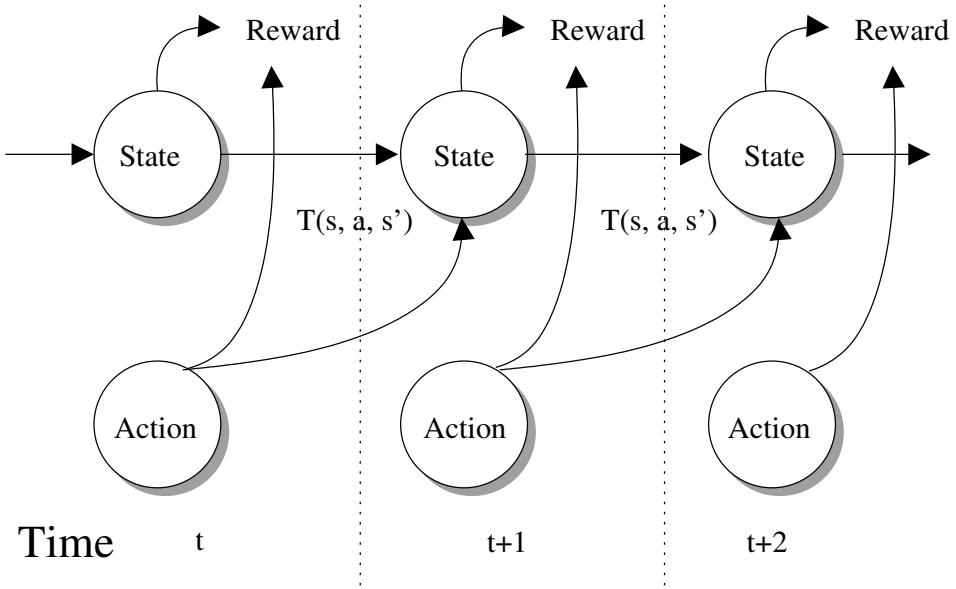
- Last Class
  - Model-Free Policy-Based Methods (Policy Gradients)
- This Class
  - Partially-Observable Markov Decision Processes
- Reference Material
  - Please follow the notes as the primary reference on this topic.
  - Planning and acting in partially observable stochastic domains, *Leslie Pack 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
  - AIMA: Ch 17 (Sections 17.4.1 - 17.4.2)

Acknowledgements

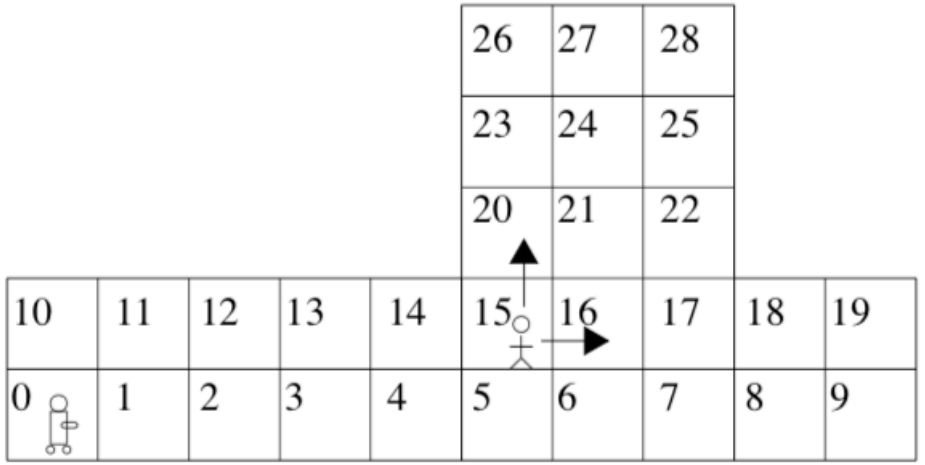
These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Nicholas Roy, Wolfram Burgard, Dieter Fox, Sebastian Thrun, Siddharth Srinivasa, Dan Klein, Pieter Abbeel, Max Likhachev, Alexander Amini (MIT Introduction to Deep Learning) and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

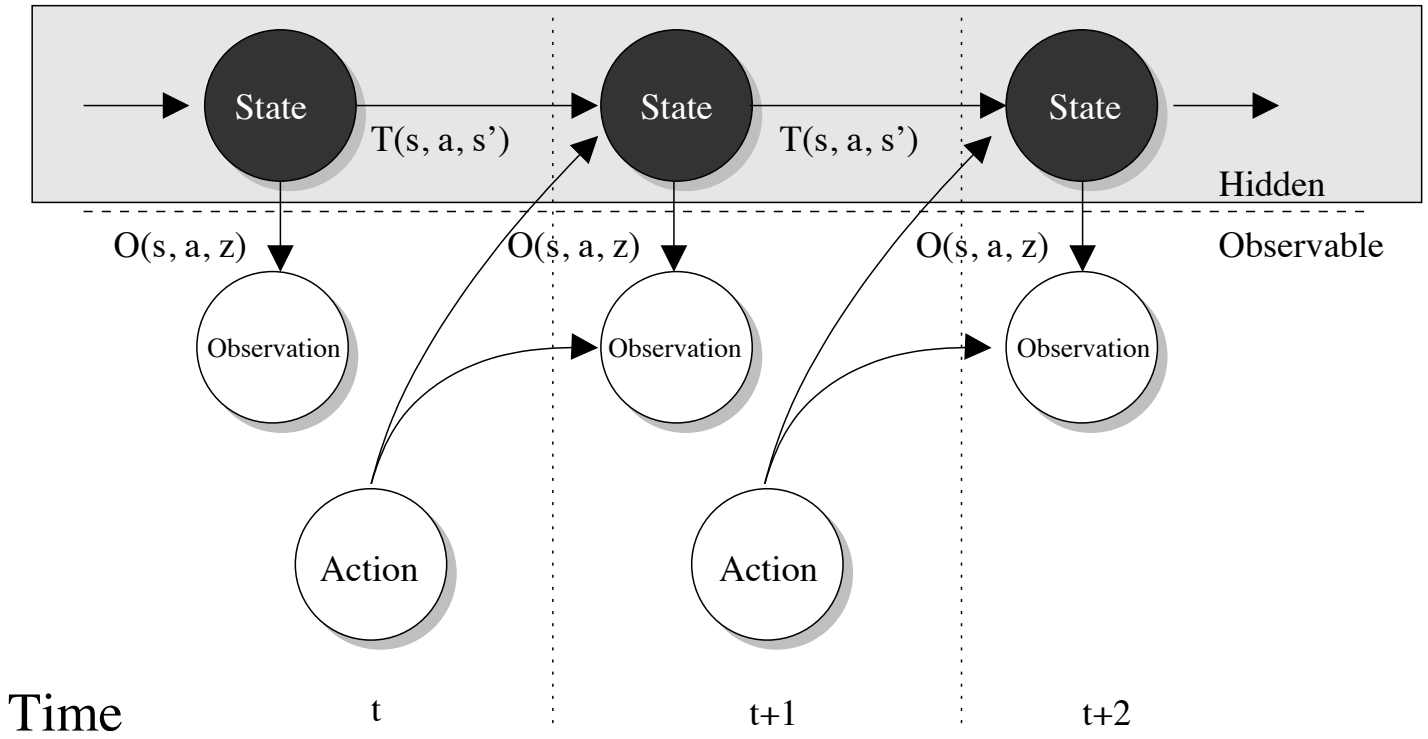
Partial Observability in the MDP Formulation

- 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

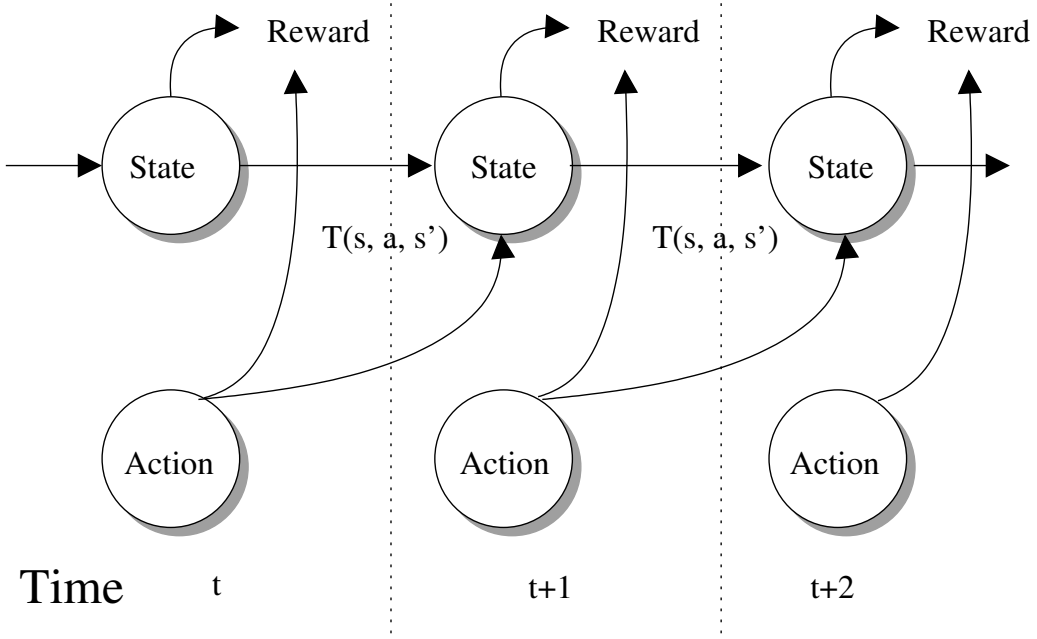
- When the true state of the world cannot be observed directly (or is "inaccessible"), the world is considered as "Partially Observable."
- In partially observable domains, decisions cannot be made based on the state of the world;
- In partially observable domains, decisions are made based on our observations 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

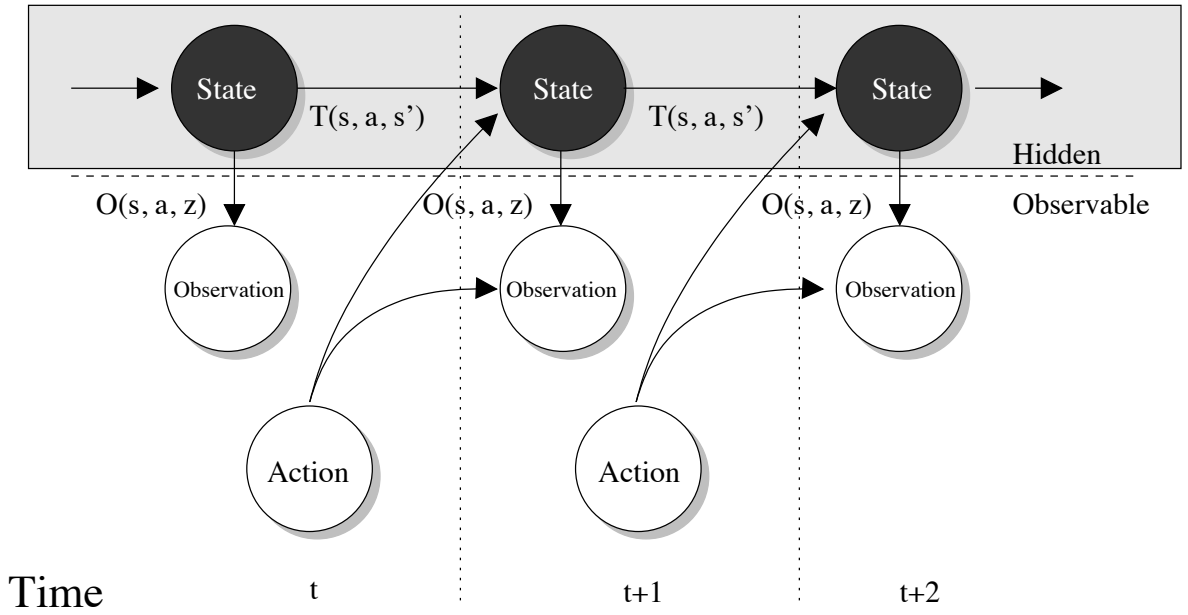


MDPs and POMDPs

MDP Graphical Model



POMDP Graphical Model



Reward not shown for brevity

MDPs and POMDPs

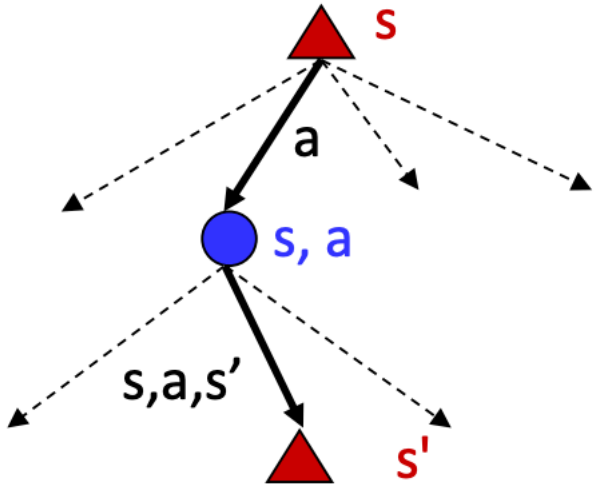
MDPs have:

States S

Actions A

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

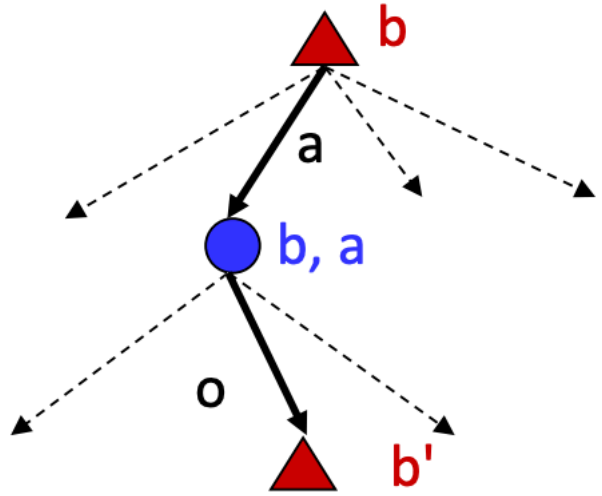
Reward $R(s,a,s')$



POMDPs add:

Observations O

Observation Function $P(o|s)$



Modeling Intent using POMDPs

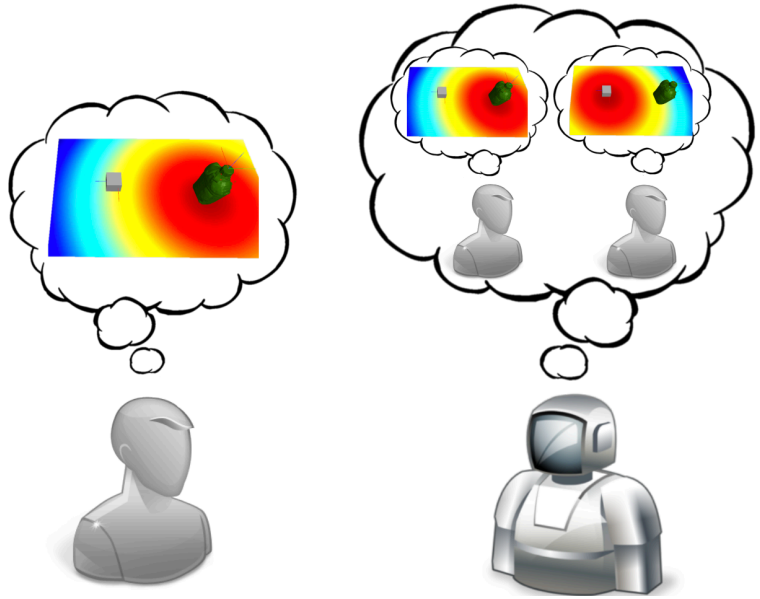


Fig. 1. Demonstration of our FETCH-POMDP model correctly fetching item for user. Note the robot's understanding of implicit information between panels three and four. This reasoning is not hard-coded into our system, but emerges from the solution of our POMDP.

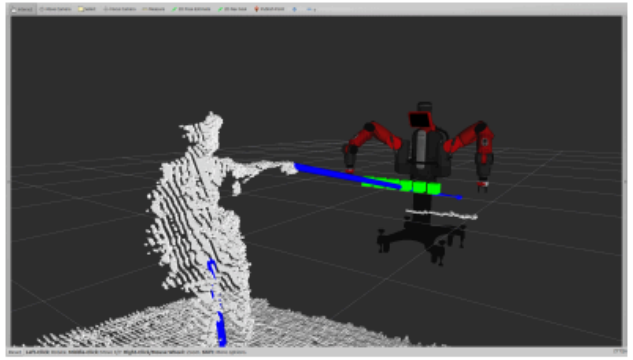


Fig. 3. Screen-capture of RViz visualization of user pointing at item. The blue vectors represent the calculated pointing vectors from each arm. The left arm is down at the user's side, and the right arm is pointing at item four.

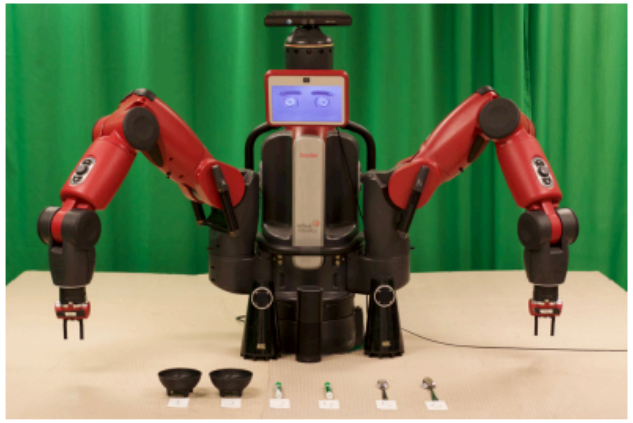


Fig. 4. User's view of robot, with items arranged in the ambiguous configuration.

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

POMDPs for Grasping

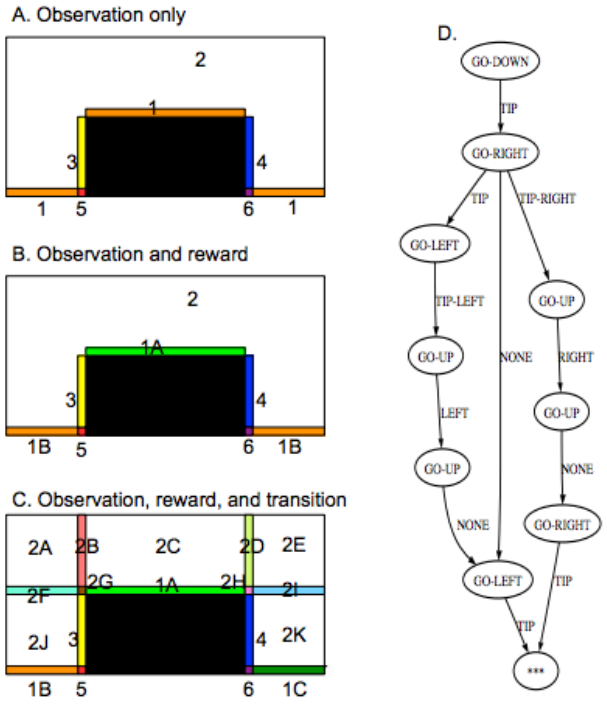


Fig. 1. A. Observation partition; B. Observation and reward partition; C. Closed partition; D. Partial policy graph for robot starting in an unknown state above the table, with a deterministic transition and observation model.

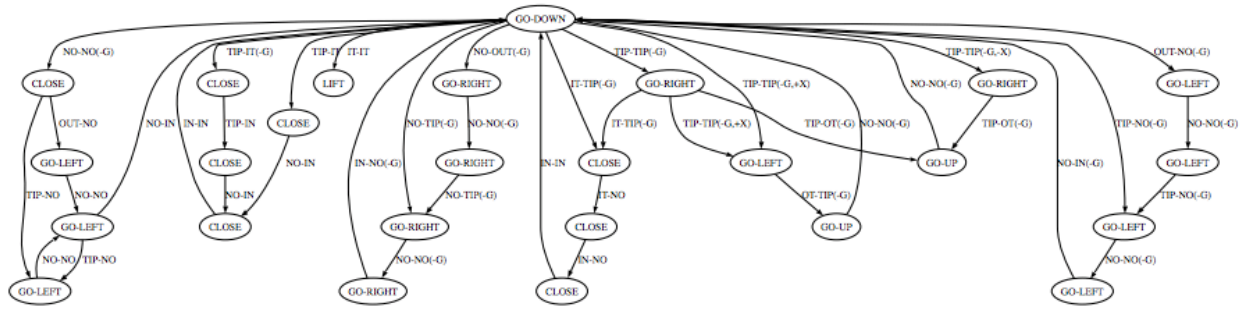


Fig. 5. Two-finger grasping policy for deterministic model.

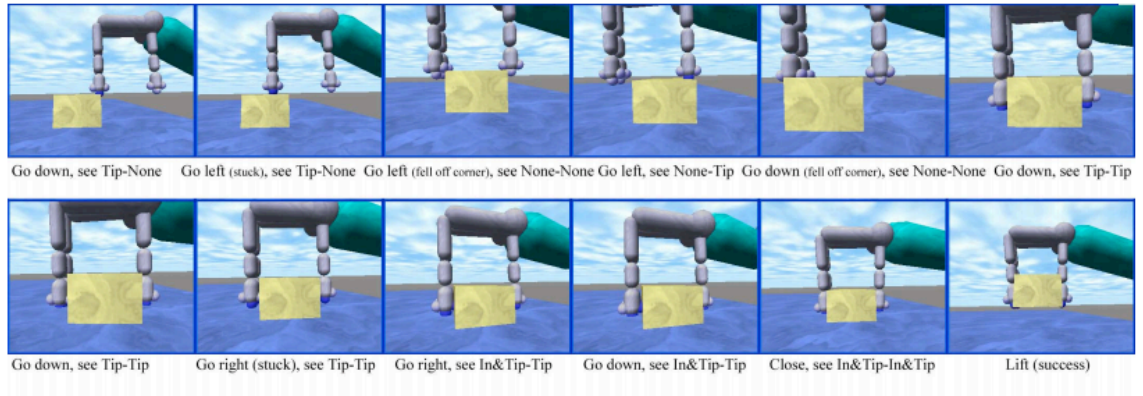


Fig. 6. Sample run of two-finger grasp policy in high-fidelity simulation.

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.

POMDP as a Belief State MDP

MDPs

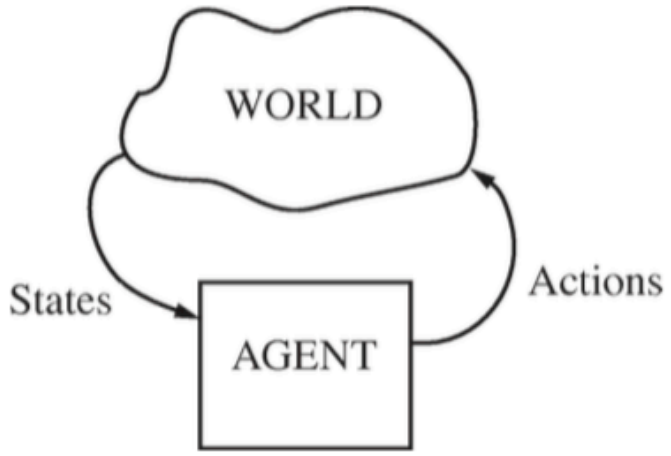


Fig. 1. An MDP models the synchronous interaction between agent and world.

POMDPs

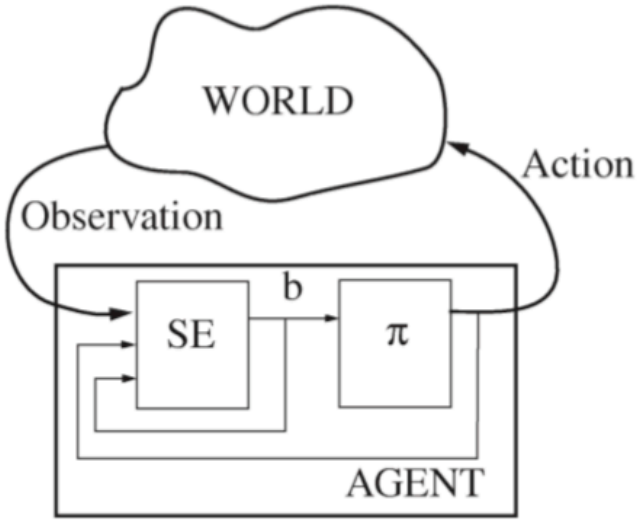


Fig. 2. A POMDP agent can be decomposed into a state estimator (SE) and a policy (π).

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.

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

- $\tau(b, a, b')$ 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

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

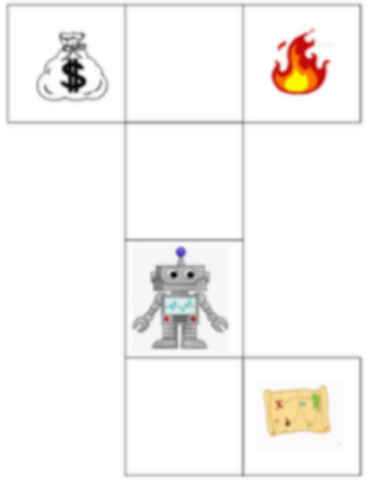
- $\rho(b, a)$ is the reward function on belief states, constructed from the original reward function on world states:

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

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,

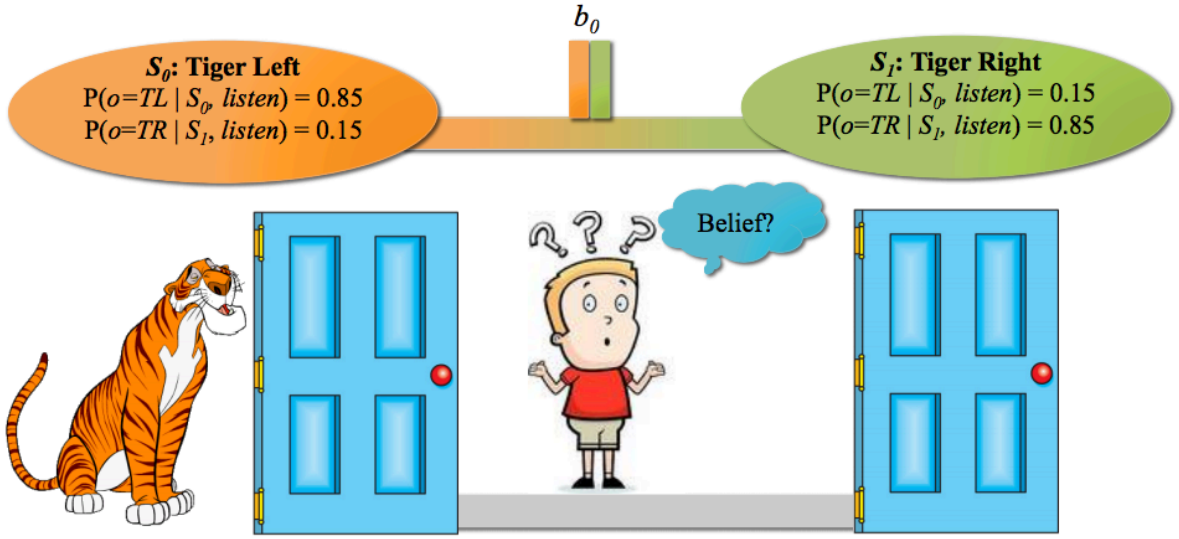
$$p(s_i|\text{listen}, s_j) = \begin{cases} 1 & \forall s_i == s_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

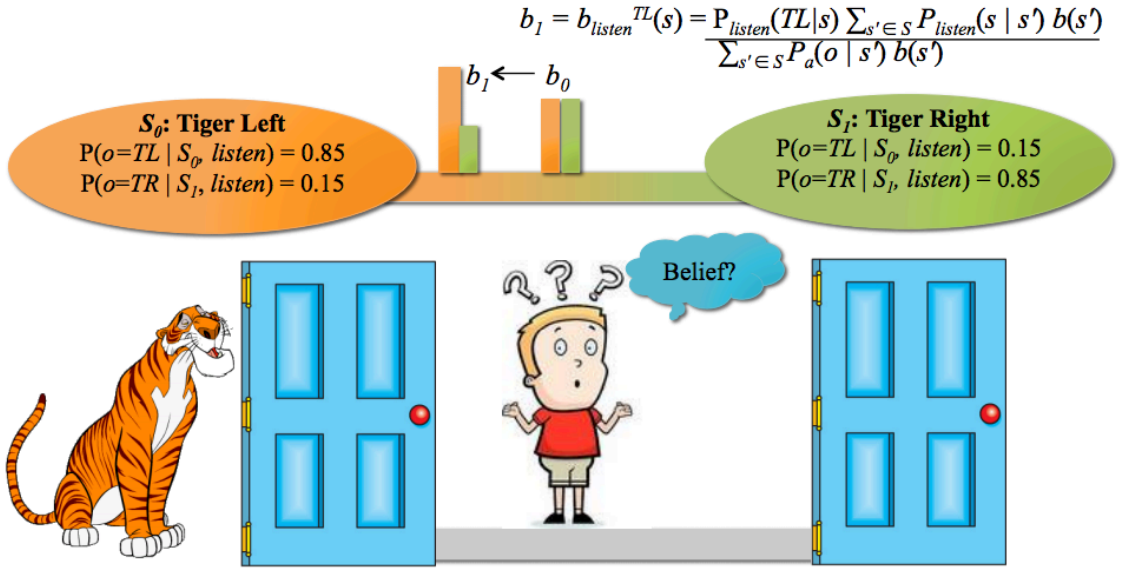
  - open-left, open-right : transition to either tiger-left or tiger-right with probability 0.5
- There are two observations: heard-left, heard-right
- The observation probabilities are as follows:
  - After the listen action:

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

  - 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

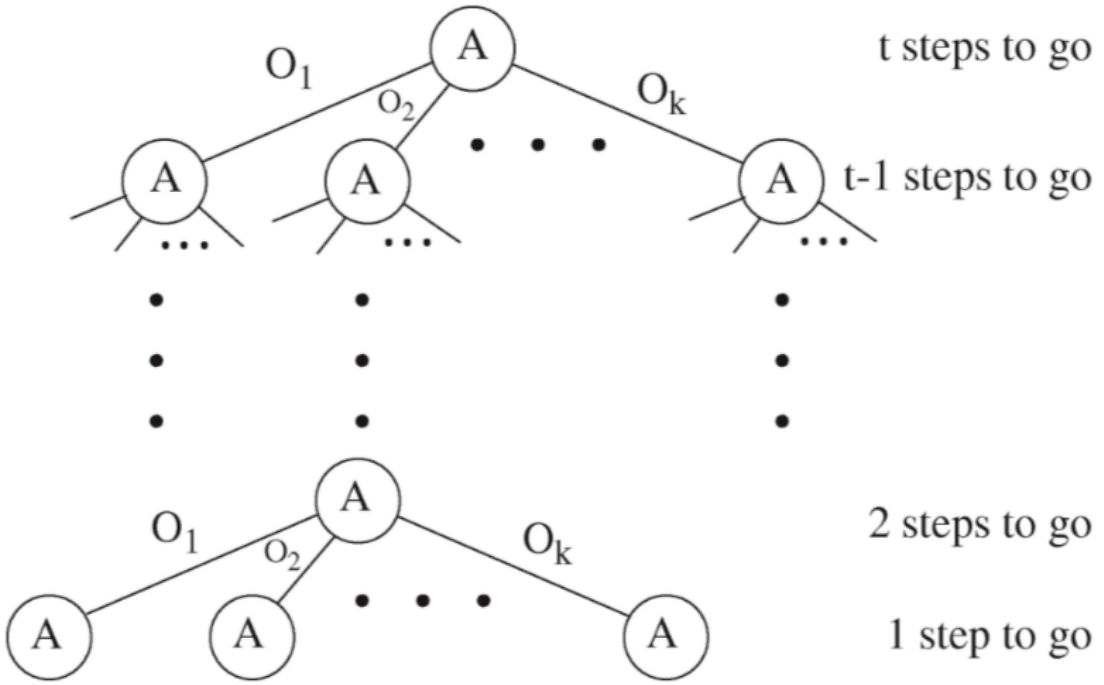




Updating the belief over the state using action and observation.
Action: Listen and Observation: Hear-Left

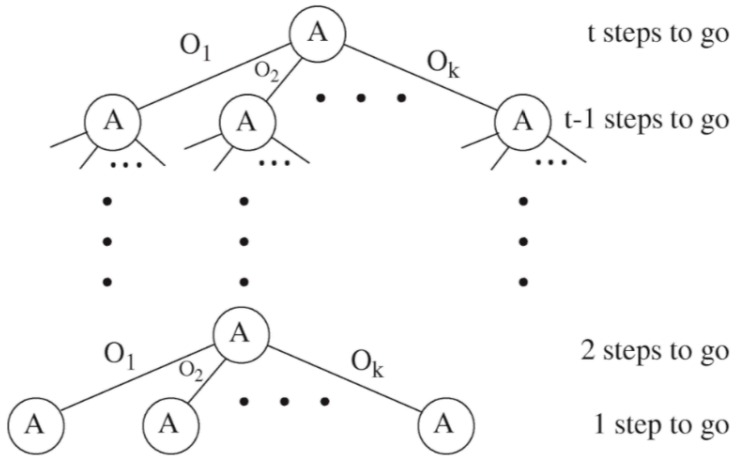
Value Function for POMDPs

- For MDPs
  - Can compute the optimal value function and then used to determine the optimal policy.
  - How to do the same for POMDPs?
- 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.
- Think of the policy tree as a “conditional” plan
  - The agent knows what to do for any observation that can arrive.



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.

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

T-step Case

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

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

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:

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

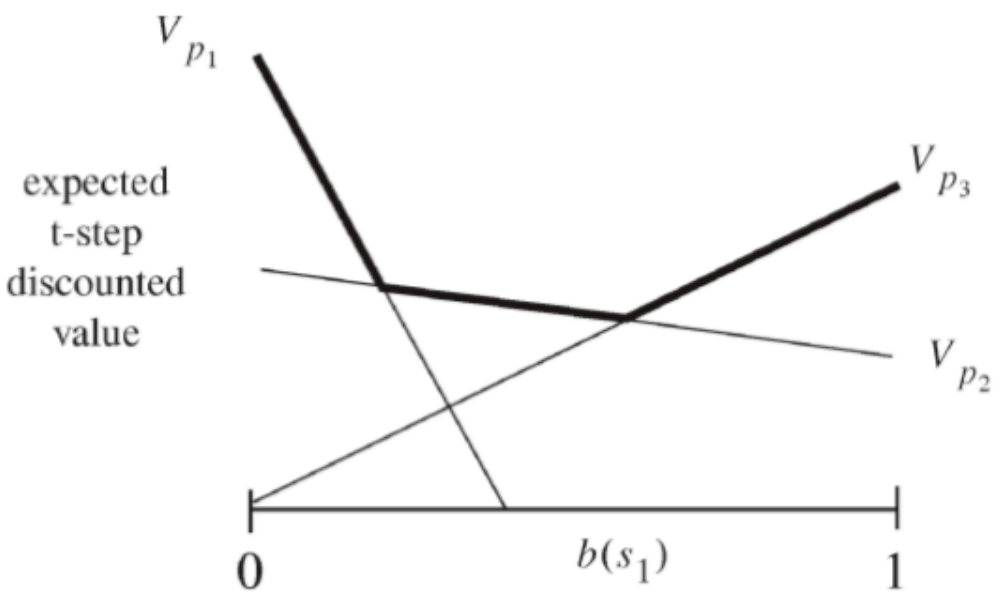
- Alpha-vector
  - Note sometimes alpha is also used to denote the value hyperplanes.

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

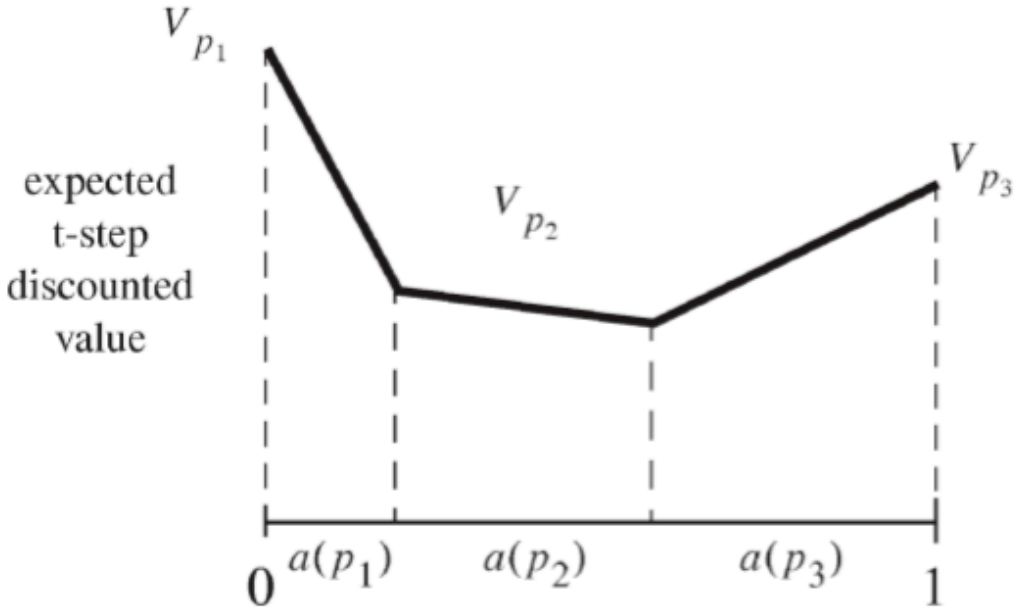
- To construct the optimal t-step policy,
  - necessary to execute different policy trees from different initial belief states.

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

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.



Optimal t-step situation-action mapping. In which region of the belief space, which policy tree is applicable.

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.

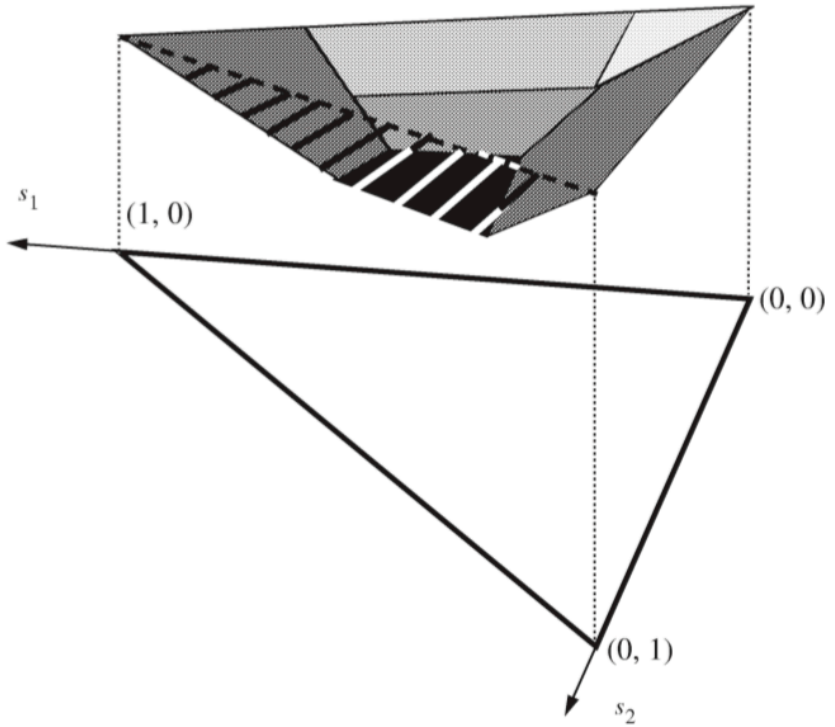
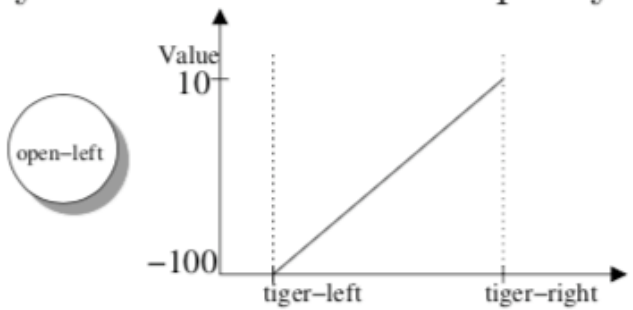


Fig. 6. A value function in three dimensions is made up of the upper surface of a set of planes.

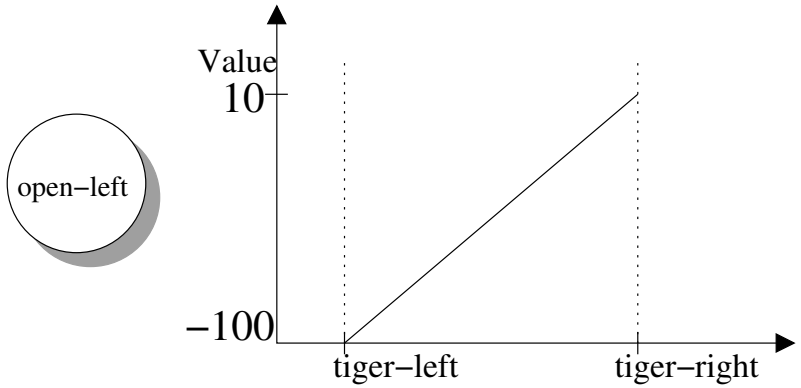
The One-Step Policy

- 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:

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

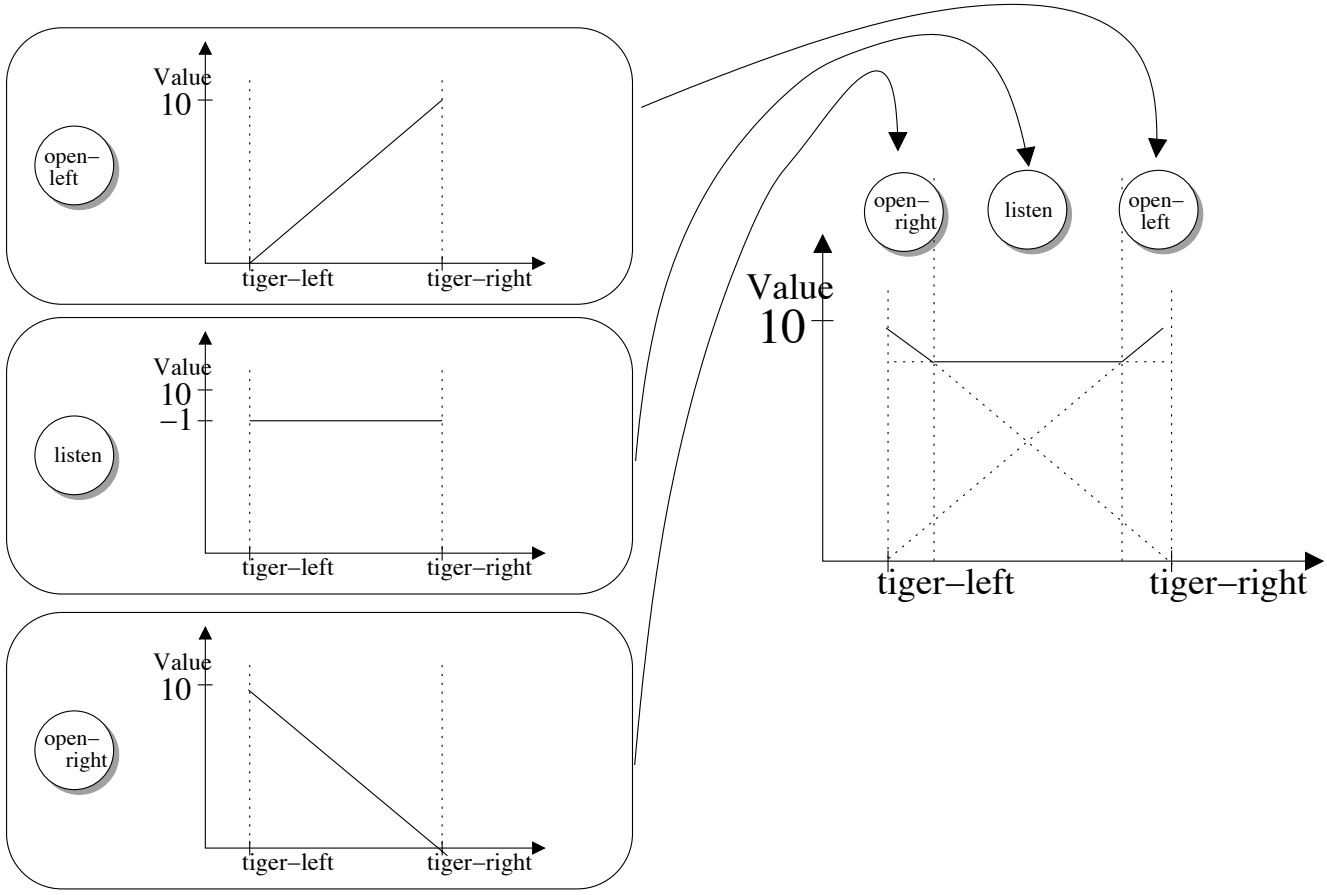
and is defined as

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

- 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.

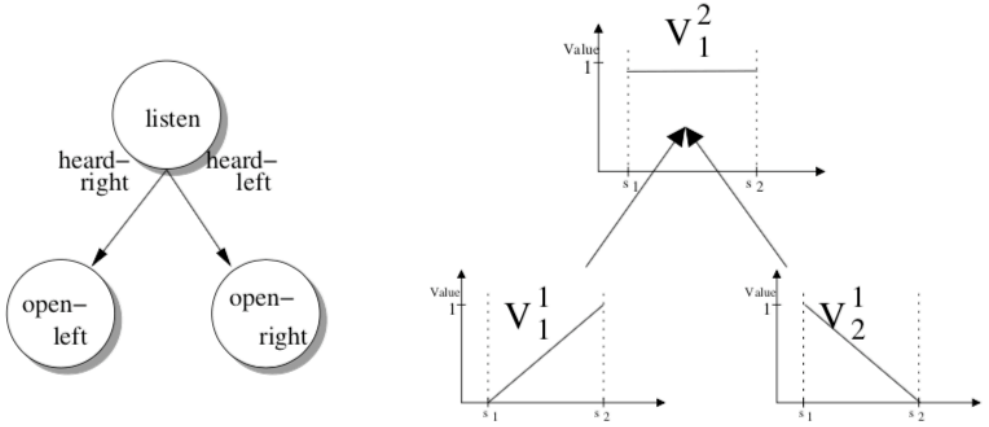
The Exact One-Step Policy

- 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.



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 z_i
    - and the probability of being at each state given the observation.

Example: Value of executing a Policy Tree

The 2-Step computation of the value of executing a policy tree rooted at the listen action. Computed for tiger-left and tiger right states. The procedure is also called backing up the alpha-vectors.

$$V_2^{\text{listen}}(\text{tiger-left}) = R(\text{tiger-left}, \text{listen}) + \gamma\Big($$
$$p(\text{tiger-left}|\text{tiger-left}, \text{listen})p(\text{heard-right}|\text{tiger-left}, \text{listen})V_1^{\text{open-left}}(\text{tiger-left}) +$$
$$p(\text{tiger-right}|\text{tiger-left}, \text{listen})p(\text{heard-right}|\text{tiger-right}, \text{listen})V_1^{\text{open-left}}(\text{tiger-right}) +$$
$$p(\text{tiger-left}|\text{tiger-left}, \text{listen})p(\text{heard-left}|\text{tiger-left}, \text{listen})V_1^{\text{open-right}}(\text{tiger-left}) +$$
$$p(\text{tiger-right}|\text{tiger-left}, \text{listen})p(\text{heard-left}|\text{tiger-right}, \text{listen})V_1^{\text{open-right}}(\text{tiger-right})\Big)$$

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

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

$$V_2^{\text{listen}}(\text{tiger-right}) = -1 + 0.75\Big(0 \times 0.15 \times -100 + 1 \times 0.85 \times 10 +$$
$$0 \times 0.85 \times 10 + 1 \times 0.15 \times -100\Big)$$

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

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.

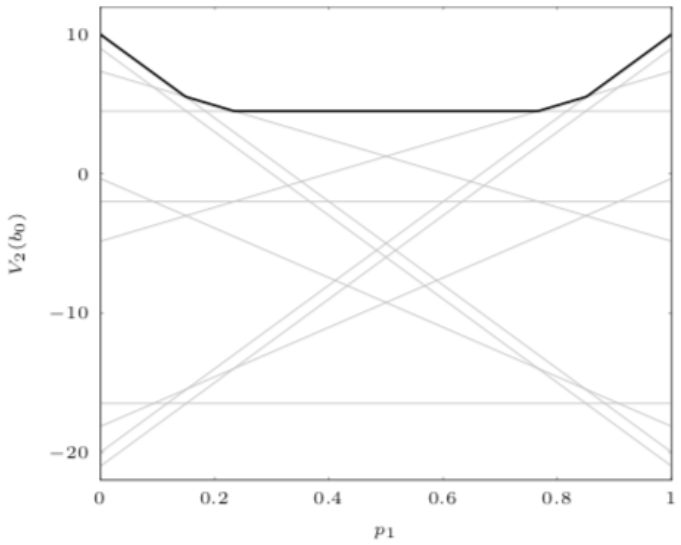
$$
\begin{aligned}
V_p(s) &= R\big(s, a(p)\big) + \gamma \cdot \big(\text{Expected value of the future}\big) \\
&= R\big(s, a(p)\big) + \gamma \sum_{s' \in \mathcal{S}} \Pr\big(s' \mid s, a(p)\big) \sum_{o_i \in \Omega} \Pr\big(o_i \mid s', a(p)\big) V_{o_i(p)}\big(s'\big) \\
&= R\big(s, a(p)\big) +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

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.

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



During value iterations, alpha vectors get refined and pruned.

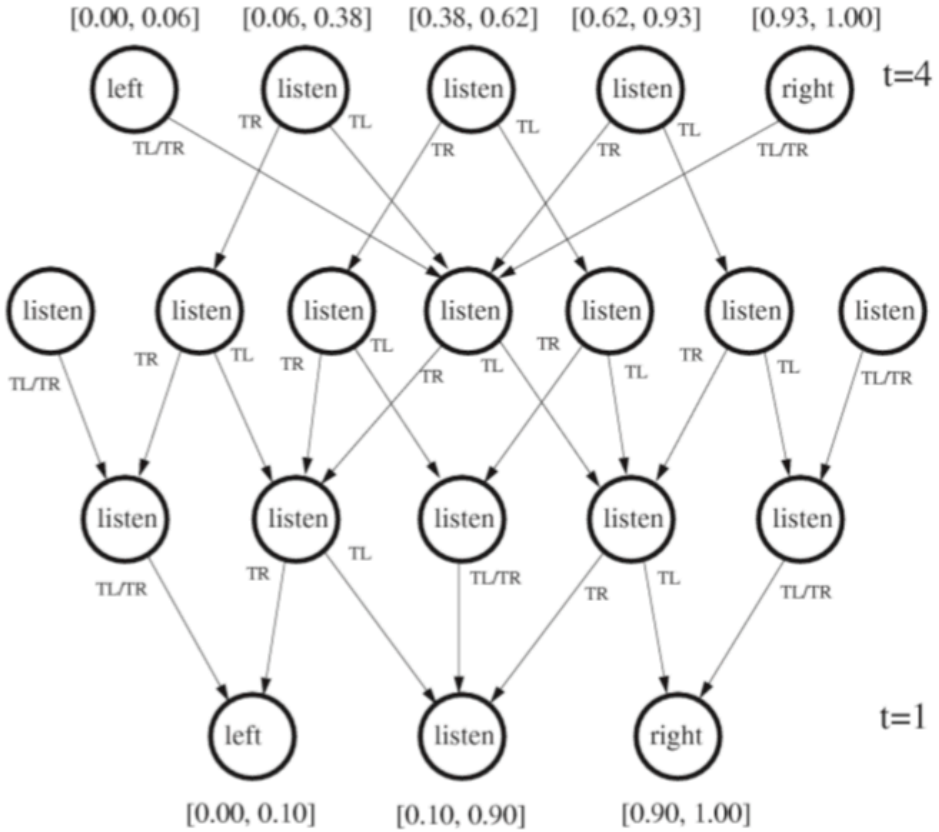
Policies for the Tiger Problem

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 timesteps

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



Other POMDP Applications

- Monitoring disease status
  - Personalized Cotesting Policies for Cervical Cancer Screening: A POMDP Approach Ebadi et al.

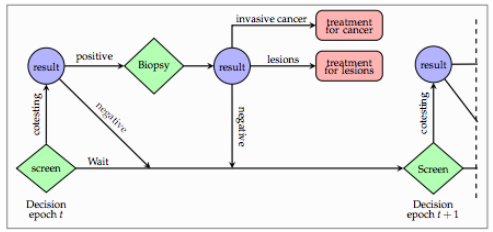


Figure 2. Timeline of the decision process.

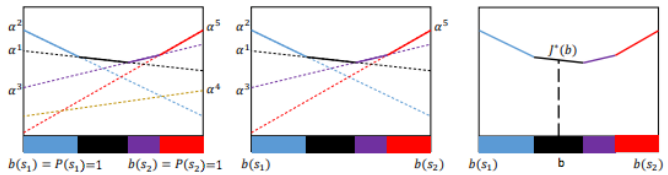


Figure 4. Alpha vectors over belief space in a two-state partially observable Markov decision process (POMDP); bold segment of the alpha vectors constitute a piecewise linear and convex (PWLC) value function. α^4 is a redundant vector and can be eliminated.

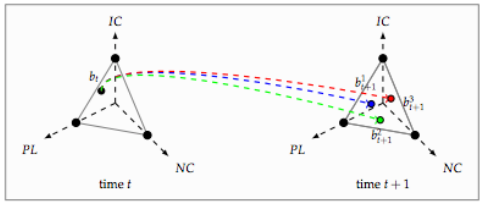


Figure 3. Belief simplex and update of belief states.

- Human-Robot Dialogue
  - Spoken Dialogue Management Using Probabilistic Reasoning, Roy et al.

Figure 1: *Florence Nightingale*, the prototype nursing home robot used in these experiments.

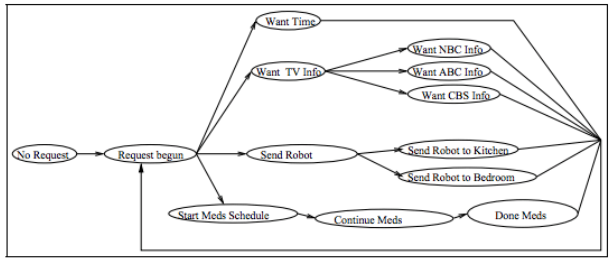


Figure 2: A simplified graph of the basic Markov Decision Process underlying the dialogue manager. Only the maximum-likelihood transitions are shown.

Observation	True State	Belief Entropy	Action	Reward
flo hello	request_begun	0.406	say_hello	100
flo what is like	start_meds	2.735	ask_repeat	-100
flo what time is it for will the	want_time	0.490	say_time	100
flo was on abc	want_tv	1.176	ask_which_station	-1
flo was on abc	want_abc	0.886	say_abc	100
flo what is on nbc	want_nbc	1.375	confirm_channel_nbc	-1
flo yes	want_nbc	0.062	say_nbc	100
flo go to the that pretty good what	send_robot	0.864	ask_robot_where	-1
flo that that hello be	send_robot_bedroom	1.839	confirm_robot_place	-1
flo the bedroom any i	send_robot_bedroom	0.194	go_to_bedroom	100
flo go it eight a hello	send_robot	1.110	ask_robot_where	-1
flo the kitchen hello	send_robot_kitchen	1.184	go_to_kitchen	100

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.