

# **CS4278/CS5478 Intelligent Robots: Algorithms and Systems**

Lin Shao

NUS

## Partially observable Markov decision process. A

(discrete) partially observable Markov decision process (POMDP) consists of the following basic elements:

- ■ •  $S$  is a set of states.
- •  $A$  is a set of actions.
- •  $Z$  is a set of observations.
- ■ •  $T(s, a, s') = p(s' | s, a)$  is a probabilistic state-transition function.
- •  $M(a, s', z) = p(z | a, s')$  is a probabilistic observation function.
- •  $R(s, a)$  is a reward function.
- •  $b_0$  is the probability distribution for the initial state.

MDP

HMM

Our objective is to find a sequence of actions  $(a_0, a_1, a_2, \dots)$  that maximize the **value** (expected total reward):

$$V = \mathbb{E} \left[ \sum_{t=0}^{N-1} R(s_t, a_t) \mid b_0 \right] \quad \text{for finite-horizon tasks,}$$

$$V = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid b_0 \right] \quad \text{for discounted infinite-horizon tasks.}$$

We solve for

- a open-loop policy  $a_t = \pi(t)$  at each time step  $t$  or
- a closed-loop policy  $a_t = \pi(b_t)$ , where  $b_t$  is the **belief**, i.e., the probability distribution over the robot state at time  $t$ .

**Question.** An MDP closed-loop policy has the form  $a_t = \pi(s_t)$ . How does the POMDP closed-loop policy differ and why?

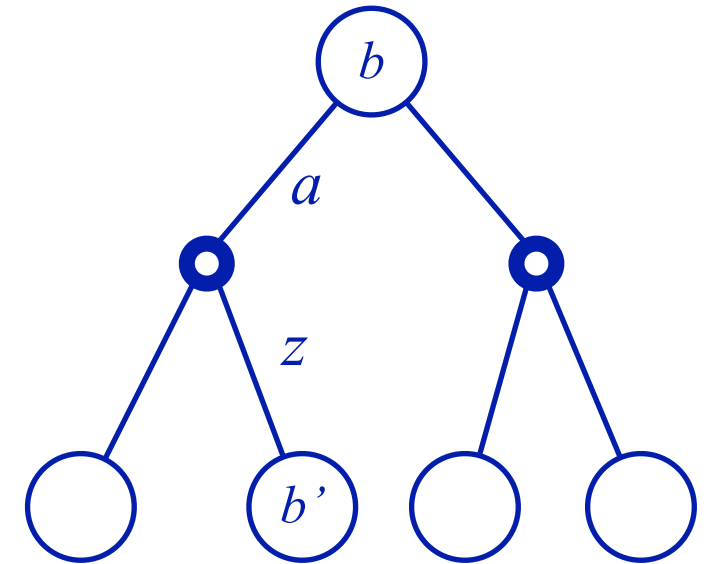
**Belief-space MDP.** To solve for a closed-loop policy, one idea is to convert a POMDP into a belief MDP. Given a POMDP, we construct an equivalent MDP with

- Each belief state  $b \in \mathcal{B}$ , where  $\mathcal{B}$  is the set of all beliefs
- Each action  $a \in A$ , where  $A$  is the set of POMDP actions
- State-transition function  $T(b, a, b')$
- Reward function  $R(b, a)$

By conditioning on  $z$ , we have

$$T(b, a, b') = p(b'|b, a) = \sum_{z \in Z} p(b'|b, a, z)p(z|b, a)$$

What is  $p(b'|b, a, z)$ ? Suppose that a robot, with initial belief  $b$ , takes action  $a$  and receives observation  $z$ . Let  $\tau(b, a, z)$  denote the new belief as a result of Bayesian filtering, i.e., prediction update with action  $a$  and observation update with observation  $z$ . If  $b' = \tau(b, a, z)$ , then  $p(b'|b, a, z) = 1$  and  $T(b, a, b') = p(z|b, a)$ . Otherwise,  $p(b'|b, a, z) = 0$ , and  $T(b, a, b') = 0$



Similarly, we calculate  $p(z|b, a)$  by conditioning and by applying the definitions:

$$p(z|b, a) = \sum_{s' \in S} p(z|b, a, s')p(s'|b, a)$$

condition on the resulting state  $s'$  after action  $a$

$$= \sum_{s' \in S} p(z|a, s') \sum_{s \in S} p(s'|s, a)p(s|b)$$

condition on the start state  $s$

$z$  is independent of  $b$ , given  $s'$

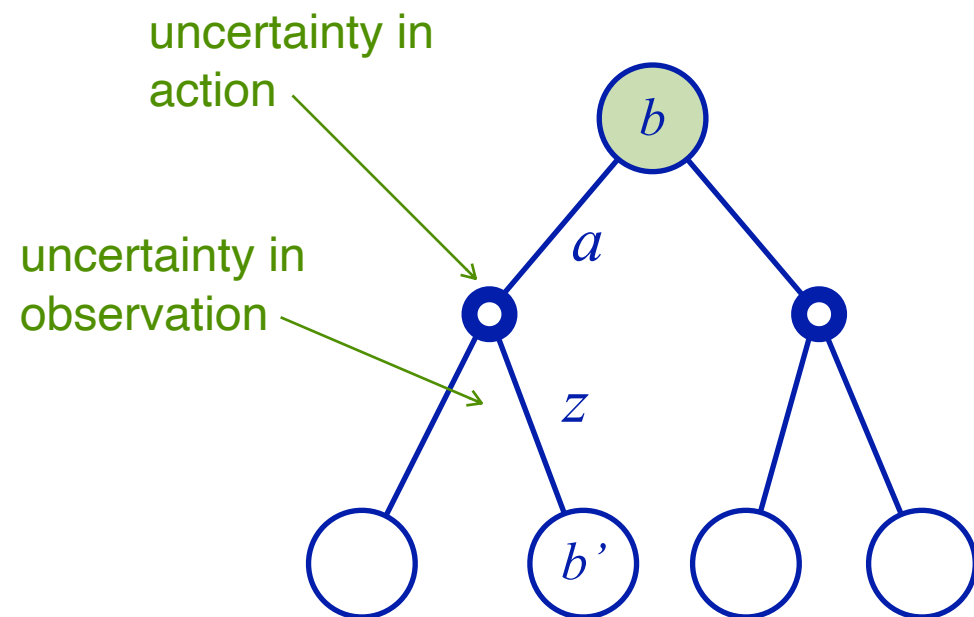
$$= \sum_{s' \in S} M(a, s', z) \sum_{s \in S} T(s, a, s')b(s)$$

definition

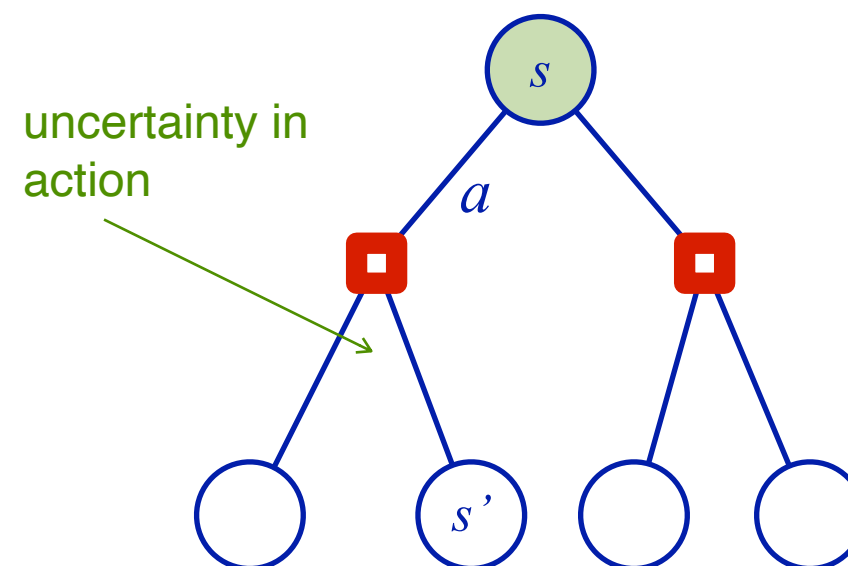
For  $R(b, a)$ , we simply average the reward  $R(s, a)$  weighted by the belief  $b$ :

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

Compare this belief MDP and the standard MDP:



Belief-space MDP



State-space MDP

Now it seems straightforward to apply value iteration:

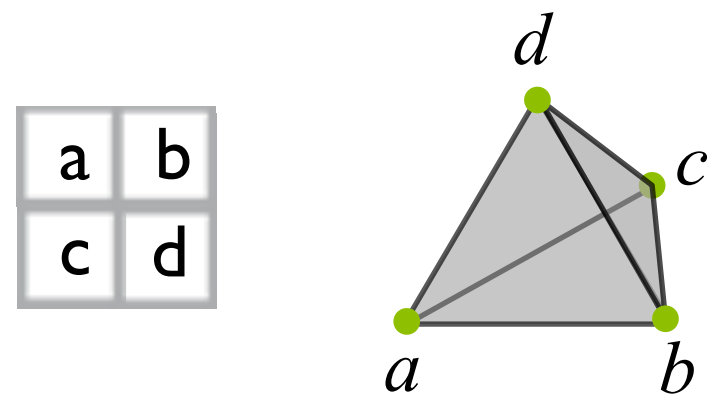
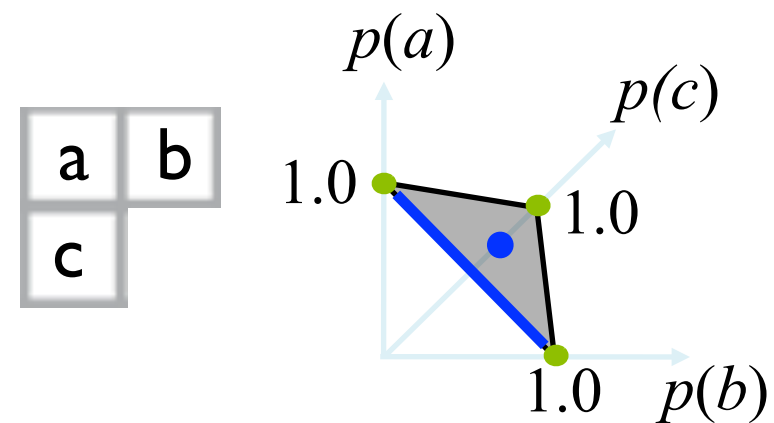
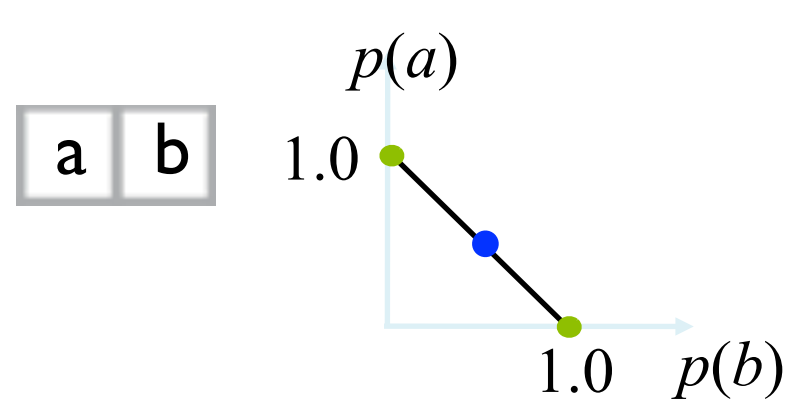
$$V_t(b) = \max_{a \in A} \left\{ R(b, a) + \gamma \sum_{b' \in \mathcal{B}} T(b, a, b') V_{t-1}(b') \right\}$$

This is conceptually correct, but is it practical algorithmically?

There are two main difficulties: policy representation and belief space size.

**Policy representation.** We can represent an MDP policy  $\pi: S \rightarrow A$  as a table, at least, when  $S$  is small. How can we represent a POMDP policy  $\pi: \mathcal{B} \rightarrow A$ , as  $\mathcal{B}$  is continuous? Maybe we should not attempt to represent  $\pi$  explicitly. We simply compute  $\pi(b)$  on the fly, given  $b$ .

**Belief space size.** Consider the geometry of  $\mathcal{B}$ .



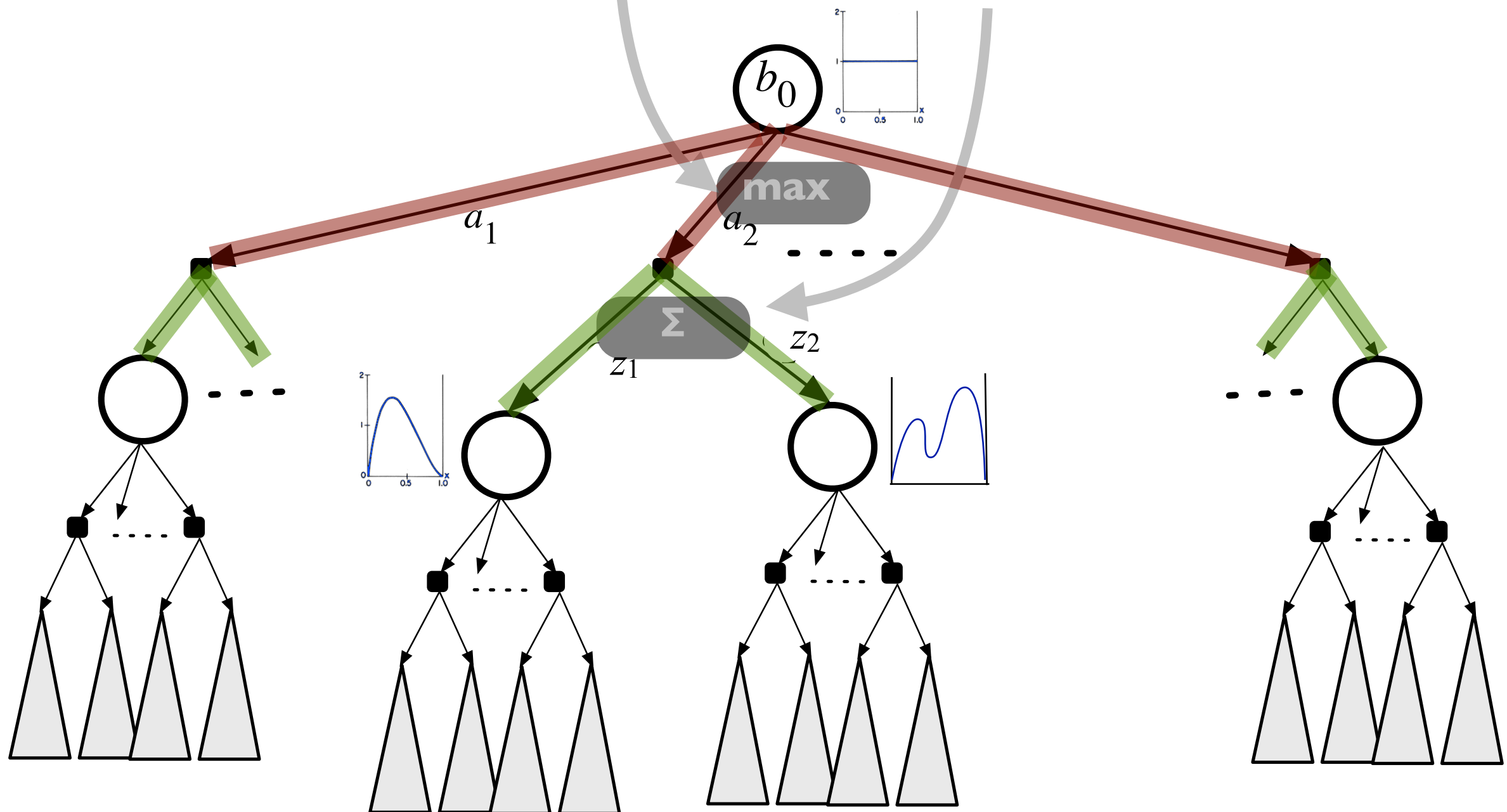
$\vdots$   
 $|S|$  states       $\vdots$   
 $(|S| - 1)$ -dim simplex

$\mathcal{B}$  seems enormous. Value iteration, which is based on the idea of dynamic programming, would never work, as it tries to process the entire space  $\mathcal{B}$ . Maybe we should consider forward search instead.

Each node of the search tree represents a belief  $b$ . At each node, the tree branches on the actions, and for each action, the tree further branches on the observations. A node  $b'$  is a child of node  $b$ , if and only if  $b' = \tau(b, a, z)$ .

same as  $\sum_{b'} T(b, a, b')$  but more compact

$$V_t(b) = \max_{a \in A} \left\{ R(b, a) + \gamma \sum_{z \in Z} p(z|b, a) V_{t-1}(b') \right\}$$





## Example (tiger).

- A girl stands in front of two closed doors. She has 3 actions: open the left door (OL), open the right door (OR), or listen (LS).
- One door leads to a tiger, thus having a large penalty (-100). The other door leads to a reward (+10). The cost of LS is -1. The discount factor is 0.95.
- There are two possible observations as a result of listening: tiger on the left (TL) or tiger on the right (TR). The observation is correct with probability 0.85.
- After a door is opened, the game resets. The tiger goes behind one of the door with equal probabilities.

Suppose that the girl has initial belief  $b$  and can take only one action.

What is her best action?

$$b = (0.5, 0.5)$$

$$b = (1, 0)$$

LS

✓  $(-1) \times 0.5 + (-1) \times 0.5 = -1$

$$(-1) \times 1 + (-1) \times 0 = -1$$

OR

$$10 \times 0.5 + (-100) \times 0.5 = -45$$

✓  $10 \times 1 + (-100) \times 0 = 10$

OL

$$-100 \times 0.5 + 10 \times 0.5 = -45$$

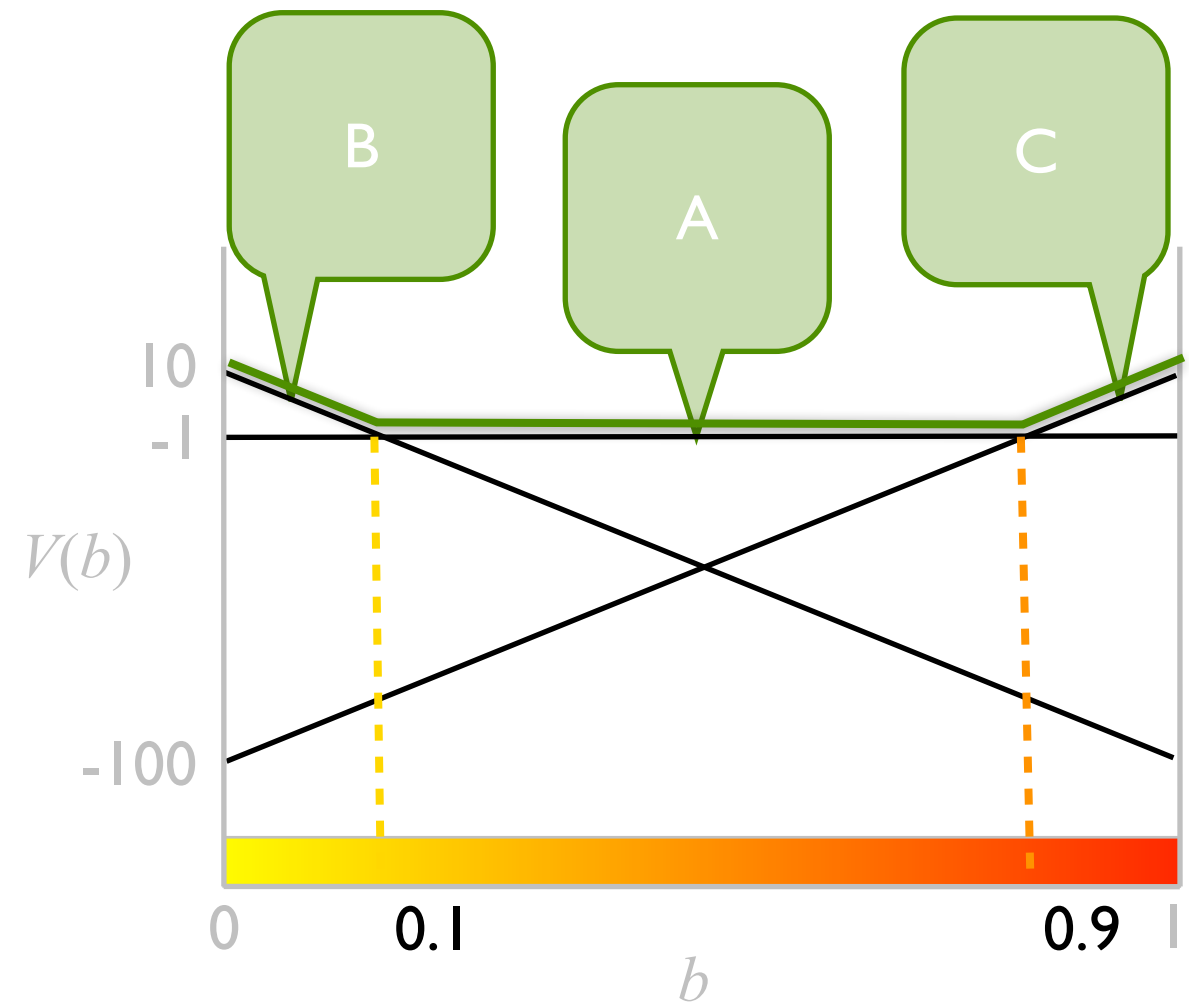
$$-100 \times 1 + 10 \times 0 = -100$$

In general, we can calculate the value of a 1-step policy with initial belief  $(b, 1 - b)$ .

A: LS  $V(b) = -1b - 1(1-b) = -1$   
 $[0.1, 0.9]$

B: OL  $V(b) = -100b + 10(1-b)$   
 $[0, 0.1]$   $= -110b + 10$

C: OR  $V(b) = 10b - 100(1-b)$   
 $[0.9, 1]$   $= 110b - 100$



**QMDP.** The QMDP algorithm drastically cuts the search depth, using a heuristic approximation to the value function  $V(b)$ . Recall

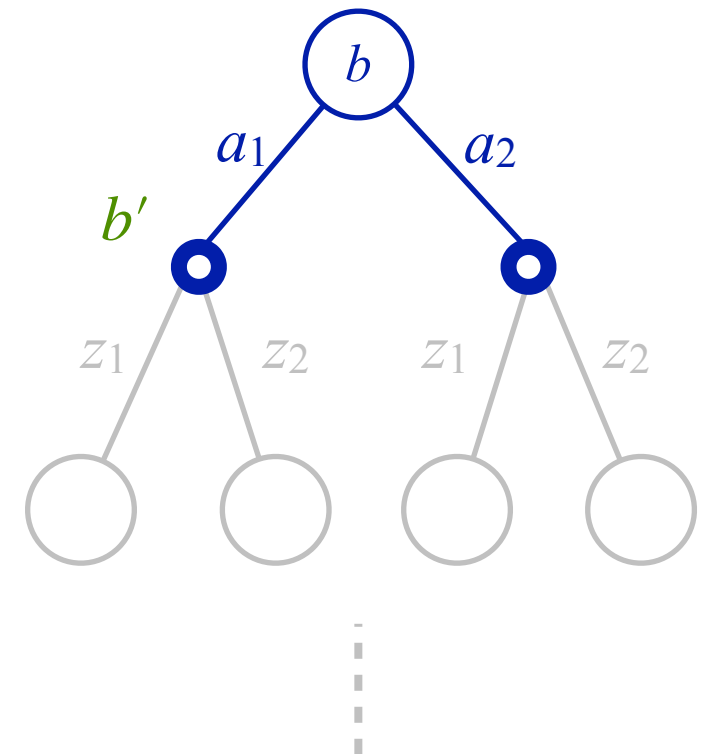
$$V(b) = \max_{a \in A} \left\{ R(b, a) + \gamma \sum_{z \in Z} p(z | b, a) V(b') \right\}$$

QMDP truncates the tree after the very first action and ignores the rest, including the first observation. We then have

$$V(b) = \max_{a \in A} \left\{ R(b, a) + h(b') \right\}$$

What is a suitable heuristic  $h(b)$ ? While it may be difficult to solve the POMDP, we can solve the corresponding MDP and obtain  $V_{\text{MDP}}(s)$ . Set  $h(b) = \sum_{s \in S} b(s) V_{\text{MDP}}(s)$ .

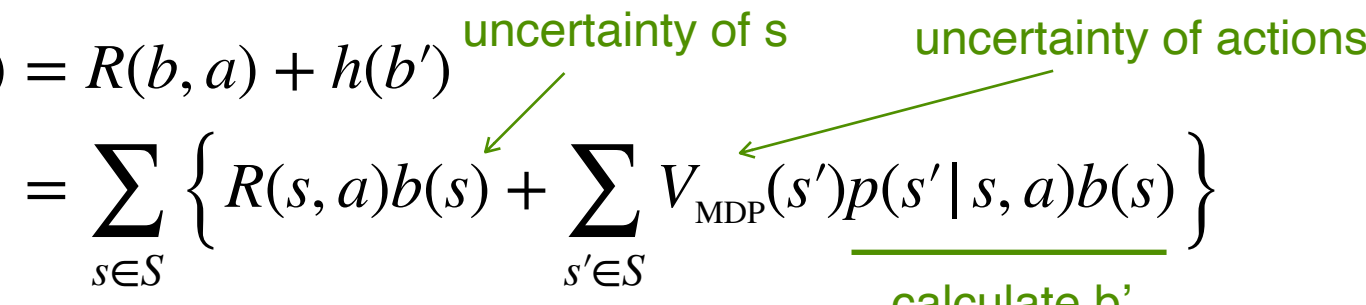
**Question.** What approximation does  $h(b)$  make? Can you say anything about the relationship between  $h(b)$  and  $V(b)$ ?



To summarize the QMDP algorithm,

- Solve the MDP for  $V_{\text{MDP}}(s)$ .
- For each action  $a \in A$ , calculate

$$\begin{aligned} Q(b, a) &= R(b, a) + h(b') \\ &= \sum_{s \in S} \left\{ R(s, a)b(s) + \sum_{s' \in S} V_{\text{MDP}}(s') \underbrace{p(s' | s, a)b(s)}_{\text{calculate } b'} \right\} \end{aligned}$$



- Finally,

$$\pi_{\text{QMDP}}(b) = \arg \max_{a \in A} Q(b, a)$$

$$V(b) = \max_{a \in A} Q(b, a)$$

QMDP reasons about the uncertainty of the current state as well as the uncertainties of actions. Effectively, it assumes that the state is fully observable after the first action and does not reason about the benefit of acquiring useful observations in the future. In this sense, it loses one main benefit of POMDP reasoning.

## Summary.

- The POMDP is a powerful tool. It reasons about uncertainty in both actions and observations. It connects robot perception and action.
- A POMDP can be converted into an equivalent belief MDP. However, the belief MDP cannot be solved using the standard MDP algorithms, such as value iteration. The belief space is enormous.
- POMDP planning is challenging, but recent algorithmic advances enables us to use it for real-time decision making in dynamic environments.