Lecture 11: Sequential Decisions with Partial Information (POMDPs)

Viliam Lisý & **Branislav Bošanský**

Artificial Intelligence Center
Department of Computer Science, Faculty of Electrical Eng.
Czech Technical University in Prague

viliam.lisy@fel.cvut.cz, **bosansky@fel.cvut.cz**

April, 2025

What we already know?

What we already covered:

- finding optimal plan
- search-based (A*) / learning-based (RL) / sampling-based (MCTS) approaches
- uncertainty

The main formal model for us was Markov Decision Process (MDP).

Unfortunately, the world is not perfect – agents often do not have perfect information about the true state of the environment
$\rightarrow$ Partially Observable MDPs (POMDPs).

Definition POMDPs

Recall the definition of POMDPs – We have a finite sets of states $\mathcal{S}$, rewards $\mathcal{R}$, and actions $\mathcal{A}$. The agent interacts with the environment in discrete steps $t = 0, 1, 2, \ldots$. At each timestep, the agent has a **belief** – a probability distribution over states that expresses the (subjective) likelihood about the current states.

The agent receives **observations** from a finite set $\mathcal{O}$ that affect the belief. The agent starts from an **initial belief** and based on actions and observations, it updates its belief. Given the current belief $b : \mathcal{S} \to [0, 1]$ and some action $a \in \mathcal{A}$ and received observation $o \in \mathcal{O}$, the new belief is defined as:

$$b(s') = \mu O(o|s', a) \cdot \sum_{s \in \mathcal{S}} Pr(s'|s, a) \cdot b(s)$$

where μ is a normalizing constant.

POMDP – Example

```
#   #   #   #   #   #
#   G               #
#   #   #   #       #
#   ↓   #   #       #
#                   #
#   #   #   #   #   #
```

The robot can now perceive only its surroundings but does not know the exact position in the maze. States and actions remain the same.

- s $= (X, Y, d, G)$
- actions = (move_forward, move_backward, turn_left, turn_right)

Observations are all possible combinations of walls / free squares in the 4-neighborhood (in front, right, behind, left):

- $(\#, \#, \#, \#), (\#, \#, \#, \_), \ldots$

Beliefs in POMDPs

So how exactly we compute the beliefs[1]:

$$a = \text{forward},\ o = (\#, \_, \_, \#)$$

$$\begin{array}{cccccc} \multicolumn{6}{c}{\text{current beliefs } b_t} \\ \# & \# & \# & \# & \# & \# \\ \# & G & 0.25 & 0.25 & & \# \\ \# & \# & \# & \# & & \# \\ \# & & \# & \# & & \# \\ \# & & 0.25 & 0.25 & & \# \\ \# & \# & \# & \# & \# & \# \end{array} \rightarrow \begin{array}{cccccc} \multicolumn{6}{c}{\text{new beliefs } b_{t+1}} \\ \# & \# & \# & \# & \# & \# \\ \# & G & & & 0.5 & \# \\ \# & \# & \# & \# & & \# \\ \# & & \# & \# & & \# \\ \# & 0.5 & & & & \# \\ \# & \# & \# & \# & \# & \# \end{array}$$

for $s' = (1, 1, <, \_)$, it holds

$$b'_{t+1}(s') = O(o|s', a) \cdot Pr(s'|a, (2, 1, <, \_)) \cdot b_t((2, 1, <, \_))$$

$$b'_{t+1}(s') = 1 \cdot 1 \cdot 0.25$$

and then $b_{t+1}(s') = \mu b'_{t+1}(s')$ where $\mu = \frac{1}{b'_{t+1}((1,1,<,\_)) + b'_{t+1}((4,4,>,\_))}$

[1] Coordinates (0,0) are in the bottom left corner.

How to act optimally in MDPs

Recall a value function for an MDP and a policy π

$$v_\pi : \mathcal{S} \to \mathbb{R}$$

is a function assigning each state s the expected return $v_\pi(s) = \mathbb{E}_\pi\, G_0$ obtained by following policy π from state s.

Optimal policies share the same **optimal state-value function:**

$$v_*(s) = \max_\pi v_\pi(s) \ \text{ for all } \ s \in \mathcal{S}$$

Any policy that is greedy with respect to v_* is an optimal policy.

$$\pi_*(s) = \arg\max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right]$$

How things change for POMDPs?

Which action is optimal depends on the **belief** over states:

#	#	#	#	#	#
#	G			$>$ (0.5)	#
#		#	#		#
#		#	#		#
#	$<$ (0.5)				#
#	#	#	#	#	#

Consider 2 actions – **move backward** and **turn right**

- **move backward** is better for the state $(4, 4, >, \_)$
- **turn right** is better for the state $(1, 1, <, \_)$

The value of each action depends on the exact belief $\rightarrow$ value function also depends on beliefs.

Value function for POMDPs

A value function for a POMDP and a policy π

$$v_\pi : \Delta(\mathcal{S}) \to \mathbb{R}$$

Can we update Bellman equation to use beliefs? Yes!

$$v_*(b) = \max_a \int p(b', r|b, a) \left[r + \gamma v_*(b')\right] db'$$

... the "only problem" is that b is a continuous variable
$\to$ computing optimal value function in this form is not practical.

Representation of Value Function

Using beliefs, we have formulated an **MDP with a continuous set of states**.

Discretization of beliefs is not very practical due to high dimension ($|\mathcal{S}|$).

Consider the Bellman equation again – what is our goal?

$$v_*(b) = \max_a \int p(b', r|b, a) \left[r + \gamma v_*(b')\right] db'$$

Find the best action (and value) for each belief point.

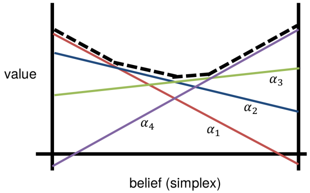
There is infinitely many belief points, but the set of actions $\mathcal{A}$ is finite!

Representation of Value Function – α vectors

If we fix an action $a \in \mathcal{A}$, the value function (for that action) is a **linear function** in the current belief. These linear functions are called α-**vectors**.

For each belief point, we take the best action hence we maximize over all α-vectors:

$$v(b) = \max_{\alpha} \sum_{s \in \mathcal{S}} \alpha(s) \cdot b(s)$$



α-vectors are in fact more general $\rightarrow$ they represent expected value for a **policy** (contingency plan consisting of multiple steps).

Using α-vectors in value iteration

Using α-vectors corresponding to the value functions of currently considered policies, we can compute new value (next iteration):

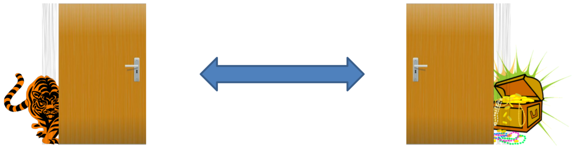
$$v_{t+1}(b) = \max_a \left\{ \sum_{o \in O} \max_{\alpha' \in v_t} \left[\sum_{r,s,s'} \mu p(s', r|s, a) b(s) O(o|s', a) \left(r + \gamma \alpha'(s')\right) \right] \right\}$$

... but how do we construct α-vectors from v_{t+1}?

1. assume there are α-vectors α' representing values of policies in step t
2. in step $t+1$, we choose some action and then, **based on the observation**, we follow with some of the policy corresponding to α' from v_t (different observation leads to a different belief)
3. for example, choose action a_3 and then
   - if o_2 is received, use value of α'_4 (i.e., this value is achievable via some policy corresponding to this α-vector)
   - if o_1 is received, use value of α'_2

Tiger example

Let's consider the best-known POMDP example – a tiger problem: There are 2 doors hiding a treasure or a tiger. The agent does not know where is the tiger and where is the treasure. The agent can gather observations (listen) or open one of the doors.



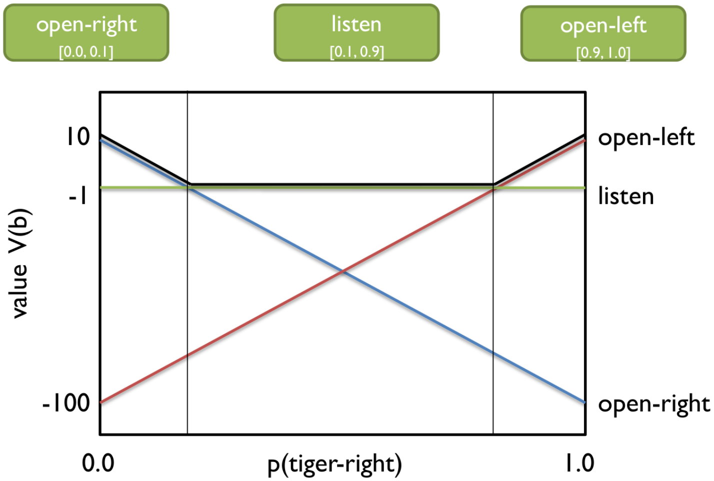
- states – $\{\text{tiger\_left}(TL), \text{tiger\_right}(TR)\}$
- actions – $\{\text{open\_left}, \text{open\_right}, \text{listen}\}$
- observations – $\{\text{hearTL}, \text{hearTR}\}$
- rewards –
  - -1 for any listening action (in all states)
  - $+10$ for opening the door with treasure
  - -100 for opening the door with tiger

Tiger example

- states – $\{\text{tiger\_left}(TL), \text{tiger\_right}(TR)\}$
- actions – $\{\text{open\_left}, \text{open\_right}, \text{listen}\}$
- observations – $\{\text{hearTL}, \text{hearTR}\}$
- rewards –
  - -1 for any listening action (in all states)
  - $+10$ for opening the door with treasure
  - -100 for opening the door with tiger
- initial belief is uniform – $b_0(TL) = b_0(TR) = 0.5$
- transition dynamics –
  - performing action **listen** does not change the state
  - opening a door "restarts" the problem (i.e., $p(s'|s,a) = 0.5$ for both states $s' \in \{TL, TR\}$).
- observation probabilities –
  - listening action generates observation **hearTL/TR** with a 15% error – i.e., agent chooses action $a =$ **listen**, then $O(\text{hearTR}|a, TR) = 0.85$ and $O(\text{hearTR}|a, TL) = 0.15$.

Tiger example

What are the optimal actions (1-step policy)?



Tiger example

Choosing action **listen** is not sufficient $\rightarrow$ what should we do next?

Depending on the observation, the belief will change:

- assume $b_0(TR) = b_0(TL) = 0.5$, $a = \text{listen}$, and $o = \text{hearTR}$
- now $b_1(TR) = \frac{0.5 \cdot 0.85}{0.5 \cdot 0.85 + 0.5 \cdot 0.15} = 0.85$

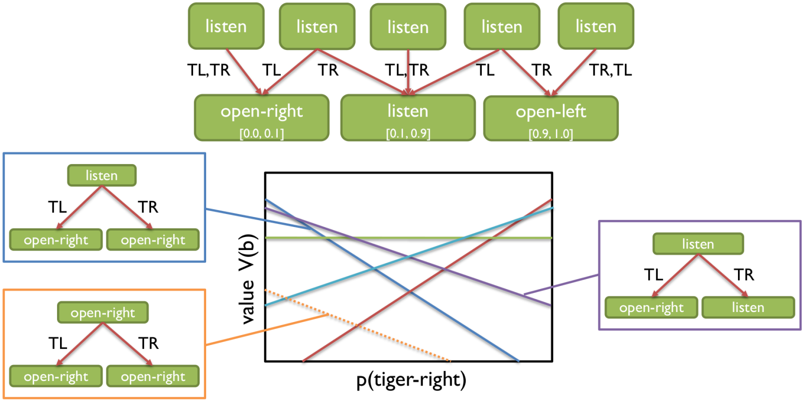
Since $0.85 \in [0.1, 0.9]$, after one observation the next optimal action is still **listen**.

In general, the chosen actions in policies depend on received observation, for example (a 2-step policy):

- **listen**
  - if (observation is hearTR $\rightarrow$ open_left)
  - else if (observation is hearTL $\rightarrow$ listen)

Tiger example

What do the α-vectors corresponding to 2-step policies look like?



Exact value iteration in POMDPs

In exact (full) value iteration in POMDPs, $|V_t| = |\mathcal{A}| \cdot |V_{t-1}|^{|\mathcal{O}|}$ new α-vectors are generated in each step of the algorithm.

It is clear that such approach will not scale well. Pruning dominated α-vectors is possible but does not solve the issue.

Observation

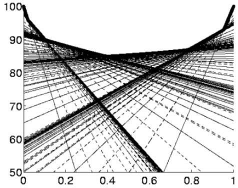
We do not need to compute all α-vectors – large portion of belief space is (often) not reached hence not relevant for solving the problem.

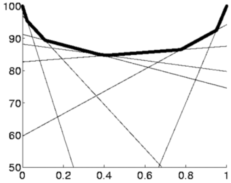
We can keep only a bounded number of belief points and for each belief point we keep 1 (the best) α-vector.

Point-based updates and point-based value iteration (PBVI)

Let $\mathcal{B} = \{b^1, b^2, \dots\}$ be a set of $|\mathcal{B}|$ belief points. **Point-based value iteration** performs Bellman update only for this limited set of belief points:

- instead of adding all α-vectors, only the α-vectors that are optimal in some of the belief points from $\mathcal{B}$ are kept,



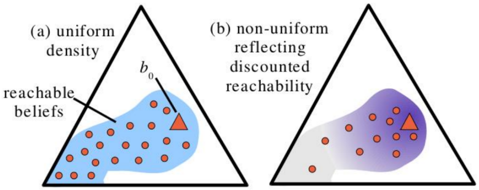


Comparison of generated α-vectors for full VI and PBVI for tiger example after 30 iterations (from slides of M. Herrmann, RL 13).

Point-based updates and point-based value iteration (PBVI)

Let $\mathcal{B} = \{b^1, b^2, \ldots\}$ be a set of $|\mathcal{B}|$ belief points. Point-based value iteration performs Bellman update only for this limited set of belief points:

- the set of belief points $\mathcal{B}$ can correspond to a uniform coverage of the belief space or the points can focus on more relevant parts of the belief space



Characteristics of PBVI

Advantages of PBVI:

- removes exponential complexity (the number of alpha vectors is bounded)
- a practical algorithm for solving POMDPs

Disadvantages of PBVI:

- it is not clear how far from the optimum is the current solution
- the set of belief points needs to be updated / maintained
- it is not clear which part of the belief space to explore

Heuristic Search Value Iteration (HSVI)

Approximates the value function with 2 approximate value functions:

- lower bound – a set of alpha vectors corresponding to infinite-step policies
- upper bound – a set of points overestimating values for each belief point

Steps of the algorithm:

1. initialization of lower bound and upper bound approximate functions
2. selecting belief points to update using a forward search (selecting the best action to explore most promising space of belief points)
3. performing point-based updates for both approximate functions

Heuristic Search Value Iteration (HSVI)

Question

How to initialize lower / upper bound value function approximations?

- lower bound – choosing some action in all belief points all the time is clearly a lower bound on the expected reward
- upper bound – solving a simplified problem $\rightarrow$ solving an MDP for each state $s \in \mathcal{S}$

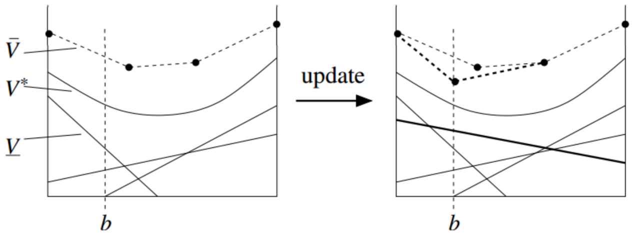
Updates:

- lower bound – point-based value updates (only the set of belief points is not bounded)
- upper bound – compute a lower convex envelope of a set of points in the upper bound and then use point-based value updates

Heuristic Search Value Iteration (HSVI)

Updates:

- lower bound – point-based value updates (only the set of belief points is not bounded)
- upper bound – compute a lower convex envelope of a set of points in the upper bound and then use point-based value updates



After an update, a new α vector is added into the lower bound and/or a new point is added into the upper bound.

Heuristic Search Value Iteration (HSVI)

Selection of the belief points to explore:

- the algorithm explores the most promising actions $\rightarrow$
  - the algorithm selects the action based on the upper bound approximation
  - see the connection with search-based methods $\rightarrow$ the upper bound is an optimistic evaluation of each belief point
  - the idea is either to (1) prove that the most-promising action actually leads to this reward (thus increase the lower bound) or (2) prove that the reward was overestimated and thus decrease the upper bound for relevant belief points
- the updates for the same action is performed for lower bound approximation

This is a very common structure of AI algorithms – upper bound drives the search, lower bounds maintains the best-found solution.

Scaling up – Monte Carlo for POMDPs

Monte Carlo Tree Search (MCTS) methods were discussed in the context of two player games.

However, we can use the same ideas for solving MDPs and also POMDPs $\rightarrow$ **POMCP** algorithm.

Question

How to construct a Monte Carlo tree?

In games (and also for MDPs), there are perfectly observable states (histories of actions) where a bandit algorithm (UCB) is used. But states are not observable in POMDPs ...

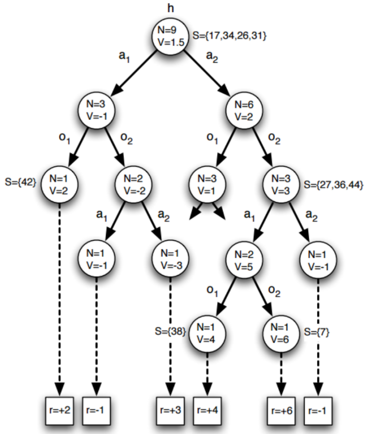
Instead, we can use **action-observation histories**.

- the agent is starting in some initial belief
- executing some action generates possible observations (according to a known probability distribution)
- receiving observation updates belief in a well-defined manner (computing a belief update)

POMCP – Monte Carlo for POMDPs

Why action-observation histories are sufficient?

- the agent is starting in some initial belief
- executing some action generates possible observations (according to a known probability distribution)
- receiving observation updates belief in a well-defined manner (computing a belief update)



POMCP – Monte Carlo for POMDPs

Where is the catch?

Bayes update of belief points can be too computationally expensive for large domains (with many states and/or observations).

The belief update can be done using **particle filtering**:

- execute K random trials (randomly choosing a true world state based on current belief, executing an action, determining the next state)
- this way we can approximate the next belief points (with probabilities of receiving an observation)

POMDPs – Key Takeaways

- Solution of the problem (strategy) is a contingency plan
  - agent needs to know which action to take in each belief, agents does not know which observation is going to be received
- Discretization via finite action sets
- Key ideas from basic models (deterministic / stochastic MDPs) work for POMDPs as well
  - The technical realization is more complex
  - Search – combination of LB best solution with valid UB estimates in HSVI
  - MCTS algorithm on action-observation histories that construct the tree