

# Partially Observable Markov Decision Process

CS4246/CS5446

Al Planning and Decision Making

This lecture will be recorded!

## **Topics**

- Partially Observable Markov Decision Process (16.4)
- Belief states and definitions
- Solution algorithms (16.5)
  - Value iteration
  - Online methods (16.5.2 and 3rd ed. 17.4.3)

# Solving Sequential Decision Problems

- Decision (Planning) Problem or Model
  - Appropriate abstraction of states, actions, uncertain effects, goals (wrt costs and values or preferences), and time horizon + observations (through sensing)
- Decision Algorithm
  - Input: a problem
  - Output: a solution as an optimal action sequence or policy over time horizon
- Decision Solution
  - An action sequence or solution from an initial state to the goal state(s)
    - An optional solution or action sequence; OR
    - An optimal policy that specifies "best" action in each state wrt to costs or values or preferences
  - (Optional) A goal state that satisfies certain properties

## Recall: Decision Making under Uncertainty

#### Decision Model:

- Actions:  $a \in A$
- Uncertain current state:  $s \in S$  with probability of reaching: P(s)
- Transition model of uncertain action outcome or effects: P(s'|s,a) probability that action a in state s reaches state s'
- Outcome of applying action a: Result(a) – random variable whose values are outcome states
- Probability of outcome state s', conditioning on that action a is executed:  $P(\text{Result}(a) = s') = \sum_{s} P(s)P(s'|s,a)$
- Preferences captured by a utility function: U(s) assigns a single number to express the desirability of a state s

# Sequential Decision Problems

- What are sequential decision problems?
  - An agent's utility depends on a sequence of decisions
  - Incorporate utilities, uncertainty, and sensing
  - Search and planning problems are special cases
  - Decision (Planning) Models:
    - Markov decision process (MDP)
    - Partially observable Markov decision process (POMDP)
    - Reinforcement learning: sequential decision making + learning

# Why Study POMDPs?

#### Uncertainty in action outcomes

- MDP: fully observable environment
- Agent knows exactly which state it is in

#### Uncertainty in observations

- POMDP: partially observable environment
- Agent does not know exactly which state it is in cannot observe the state directly
- Some noisy observations projected from a state

#### Real-world challenges

- Model sequential decision problem for an uncertain, dynamic, partially observable environment
- If the state is not directly observable, how does an agent reason about its decisions?

# Partially Observable Markov Decision Process (POMDP)

- An POMDP  $M \triangleq (S, A, E, T, O, R)$  consists of
- A set S of states
- A set A of actions
- A set *E* of evidences or observations or percepts
- A transition function  $T: S \times A \times S \rightarrow [0,1]$  such that:

$$\forall s \in S, \forall a \in A: \sum_{s' \in S} T(s, a, s') = \sum_{s' \in S} P(s'|s, a) = 1$$

• An observation function  $0: S \times E \rightarrow [0, 1]$  such that:

$$\forall s \in S, \forall e \in E: \sum_{e \in E} O(s, e) = \sum_{e \in E} P(e|s) = 1$$

- A reward function  $R: S \to \Re$  or  $R: S \times A \times S \to \Re$
- Solution: What is a policy in POMDP?

### Observation Function and Sensor Model

#### Observation function:

- O(s,e) = P(e|s) is the probability of observing (or perceiving evidence) e from state s
- Define O(s,e) for all  $s \in S$  and  $e \in E$
- Assumption of Markov property

#### Sensor model

- The observation function defines the sensor model
- An observation is also called a measurement or test

# Evidence – Real-life Examples



**Robotics**: grasping an item in a clutter evidence by pixels



**Navigation**: driving from A to B evidence by road signs/buildings/etc



**Medicine**: detecting a diagnosis of a patient *evidence by symptoms* 

# Belief State as Probability Distribution

#### Belief State:

- Actual state of the system is unknown, but we can track the probability distribution or belief state over the possible states
- An action  $\alpha$  changes the belief state b, not just the physical state s
- b(s) denotes probability assigned to actual state s by belief state b
- If b(s) is the current belief, the agent executes action a and receives evidence e, then the updated belief is given by filtering:

$$b'(s') = \alpha P(e'|s') \sum_{s} P(s'|s,a) b(s)$$

where  $\alpha$  is the normalizing constant that makes the belief state sum to 1

• Filtering Function:

$$b' = FORWARD(b, a, e')$$

### Exercise

$$b'(s') = \alpha P(e'|s') \sum_{s} P(s'|s, a)b(s)$$

- Consider a problem with two states  $s_1$  and  $s_2$ :
  - Current belief  $b(s_1) = 0.6$  and  $b(s_2) = 0.4$ .
- For action a:
  - Let the transition probabilities be:  $P(s_1|s_1,a)=0.2$  and  $P(s_2|s_1,a)=0.8$ ;  $P(s_2|s_2,a)=0.3$  and  $P(s_1|s_2,a)=0.7$
  - Let the observation probabilities be:  $P(o_2|s_1) = 0.7$  and  $P(o_2|s_2) = 0.2$
- Assume that evidence  $e' = o_2$  is received. What is b'?

$$b'(s_1) = \alpha P(o_2|s_1)[P(s_1|s_1, a)b(s_1) + P(s_1|s_2, a)b(s_2)]$$

$$= \alpha 0.7[0.2 \times 0.6 + 0.7 \times 0.4] = ?$$

$$b'(s_2) = \alpha P(o_2|s_2)[P(s_2|s_1, a)b(s_1) + P(s_2|s_2, a)b(s_2)]$$

$$= \alpha 0.2[0.8 \times 0.6 + 0.3 \times 0.4] = ?$$

# Decision Making in POMDPs

#### Main ideas:

- Optimal action depends only on agent's current belief
- Optimal policy can be described by a mapping  $\pi^*(b)$  from belief to action
- Optimal policy does not depend on the actual state agent is in!
- Include value of information as a component in decision making

#### Decision cycle of a POMDP agent:

- Given current belief b, execute action  $a = \pi^*(b)$
- Receive percept e'
- Set belief to b' = FORWARD(b, a, e') and repeat

# Example: Navigation in Grid World

#### • POMDP:

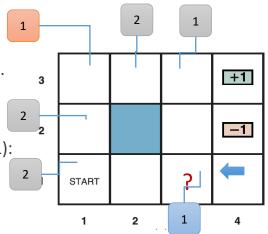
• States S, actions A, transition T, reward R, and observation model O.

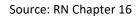
#### Assume:

- An observation function measures no. of adjacent walls in a state s
- For all non-terminal states except those in column 3 (where value = 1):
- O(s,2) = 0.9, O(s,\*) = 0.1 (where \* indicates a wrong value)

#### Belief state:

- b((1,1)) = 1/9, b((1,2)) = 1/9, ..., b((3,4)) = 0
- Suppose agent moves Left and sensor reports 1 adjacent wall
  - It is quite likely that agent is now in (3, 1)
- What is the exact probability values of the new belief state?





# Calculating New Belief State

- What is the probability that an agent in belief state b reaches belief state b' after executing action a?
  - With current belief b(s), agent executes action a and perceives e', updated belief is:

$$b'(s') = \alpha P(e'|s') \sum_{s} P(s'|s,a)b(s) = \text{FORWARD}(b,a,e')$$

where  $\alpha$  is normalizing constant for belief state to sum to 1

- New belief b' is a conditional probability over actual state given sequence of percepts and actions so far
- If action and subsequent percept were known, deterministic update to the belief using b' = FORWARD(b, a, e')
- But subsequent percept is not yet known, so agent might arrive in one of several possible belief states b', depending on percept received

# Calculating Probability of Percept

#### Probability of percept

• Probability of perceiving e', given that a was performed starting in belief state b, is given by summing over all the actual states s' that the agent might reach:

$$P(e'|a,b) = \sum_{s'} P(e'|a,s',b)P(s'|a,b)$$

$$= \sum_{s'} P(e'|s')P(s'|a,b)$$

$$= \sum_{s} P(e'|s')\sum_{s} P(s'|s,a)b(s)$$
Sensor model

Transition model

## Calculating Transition Model and Reward Model

- Probability of reaching b' from b, given action a:
  - Transition model for belief state space:

$$P(b'|a,b) = \sum_{e'} P(b'|e',a,b) P(e'|a,b)$$

$$= \sum_{e'} P(b'|e',a,b) \sum_{s'} P(e'|s') \sum_{s} P(s'|s,a) b(s).$$
Sensor model

Transition model

where P(b'|e', a, b) is 1 if b' = FORWARD(b, a, e') and 0 otherwise

- Reward function of belief state space:
  - Expected reward if the agent does *a* in belief state *b*:

$$\rho(b,a) = \sum_{s} b(s) \sum_{s'} P(s'|s,a) R(s,a,s')$$

# Reducing POMDP into an MDP

#### Belief space MDP:

- Transition model P(b'|b,a) and reward model  $\rho(b,a)$  define an observable MDP on the space of belief states!
- Solving a POMDP on a physical state space solving a continuous, usually high-dimensional MDP on the corresponding belief-state space
- An optimal policy for this MDP,  $\pi^*(b)$  is also an optimal policy for the original POMDP

#### • Remember:

The belief state is always observable to the agent, by definition

## Value Iteration

## Value Iteration for POMPDs

#### Policy and conditional plan

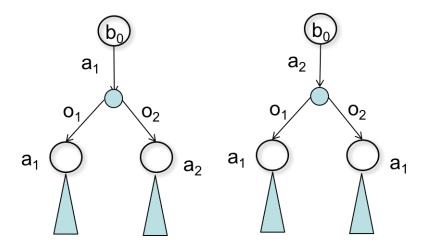
- ullet Consider an optimal policy  $\pi^*$  and its application in a single belief state b
- Policy generates an action, then for each subsequent observation or percept, belief state is updated and new action is generated, and so on
- For b, the policy is equivalent to a conditional plan

#### Main issue:

 How does the expected utility of executing a fixed conditional plan varies with the initial belief state?

## Conditional Plan

• A policy at a belief  $b_0$  is a conditional plan



• Multiple conditional plans are possible

# Utility Function for Belief State

#### Note:

- A belief state b is a probability distribution
- Each utility value in a POMDP is a function of an entire probability distribution

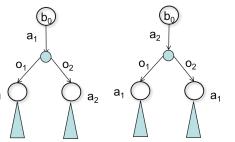
#### • Problems:

- Probability distributions are continuous
- Huge complexity of belief spaces

#### Solution:

• For finite state, action, and observation spaces and planning horizon: utility functions represented by piecewise linear functions over belief space

# Utility Function of Belief State (1) (1) (1)



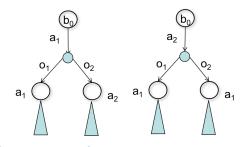
• Let the utility of executing a fixed conditional plan p starting in physical state s be  $\alpha_p(s)$  – alpha vector

• Expected utility of executing p in belief state b is:

$$U_p(b) = \sum_S b(S) \alpha_p(S)$$
 or  $b \cdot \alpha_p$  (inner product of vectors  $b$ ,  $\alpha_p$ )

• A linear function of b, corresponding to a hyperplane in belief space

# 



 At any particular belief state b, optimal policy is to choose the conditional plan with highest utility:

$$U^{\pi^*}(b) = U(b) = \max_p b \cdot \alpha_p$$

- Utility or value function U(b) on belief states, being the maximum of a collection of hyperplanes, will be piecewise linear and convex
- For finite depth, there are only a finite number of conditional plans
  - With |A| actions, |E| observations, there are  $|A|^{O(|E|^{d-1})}$  distinct depth-d plans

# Example: A Two-State World

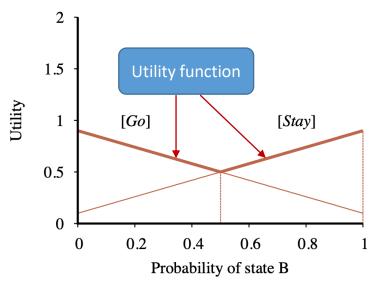
- Given:
  - Two states: A, B (or 0, 1 in RN 3e) [Belief space is 1-dimensional]
  - Rewards: R(.,.,A) = 0, R(.,.,B) = 1 [Any transition ending in A is 0, ...]
  - Two actions:
    - Stay Stays put with probability = 0.9
    - Go Switches to the other state with probability = 0.9
  - Discount factor:  $\gamma = 1$
  - Sensor: Reports correct state with probability = 0.6
- Obviously:
  - Agent should: Stay when it thinks it is in state B and Go when it thinks it is in state A
- What are the values for 1-step plans ( $\alpha$ -vectors)?

# Example: Solving a POMDP, d=1

- Consider one-step plans: [Stay] and [Go]
  - Each receives reward for one transition as follows:

$$\alpha_{[Stay]}(A) = 0.9 R(A, Stay, A) + 0.1 R(A, Stay, B) = 0.1$$
  
 $\alpha_{[Stay]}(B) = 0.1 R(B, Stay, A) + 0.9 R(B, Stay, B) = 0.9$   
 $\alpha_{[Go]}(A) = 0.1 R(A, Go, A) + 0.9 R(A, Go, B) = 0.9$   
 $\alpha_{[Go]}(B) = 0.9 R(B, Go, A) + 0.1 R(B, Go, B) = 0.1$ 

- Hyperplanes for  $b \cdot \alpha_{[Stay]}$  and  $b \cdot \alpha_{[Go]}$ 
  - Utility or value function: max of the two linear functions
- What is the one-step optimal policy?
  - Optimal policy is to Stay if b(B) > 0.5 and Go otherwise



Utility of Belief-State b(B): d = 1

Source: RN Figure 17.15 (a)

# Example: Solving a POMDP, d=2

- Deriving a solution:
  - Obtain utilities  $\alpha_p(s)$  for all the conditional plans p of depth 1 in each physical state s
  - Compute utilities for conditional plans of depth 2 by considering:
    - each possible first action
    - · each possible subsequent percept, and
    - each way of choosing a depth-1 plan to execute for each percept:

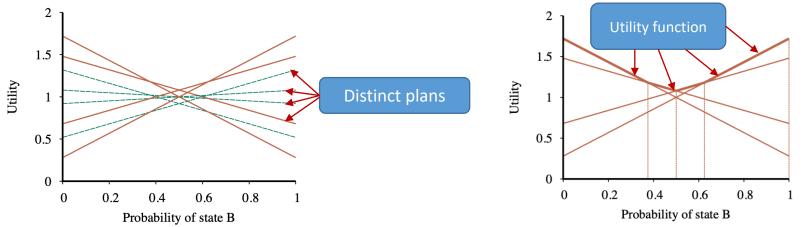
```
[Stay; if Percept = A then Stay else Stay]

[Stay; if Percept = A then Stay else Go]

[Go; if Percept = A then Stay else Stay] ...
```

- There are 8 distinct depth-2 plans in all
  - 4 suboptimal and dominated plans dominated plans are never optimal
  - 4 undominated plans, each optimal in a specific region
- The regions partition the belief-state space

# Example: Utility of Belief-State b(B): d=2



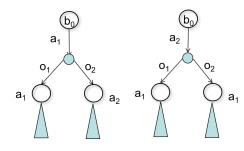
Utilities of 8 distinct 2-step plans

Utilities of 4 undominated 2-step plans

- Continuous belief space is divided into regions; the regions partition the belief-state space
  - Each region corresponds to a conditional plan that is optimal for that region
- U(b) is piecewise linear and convex

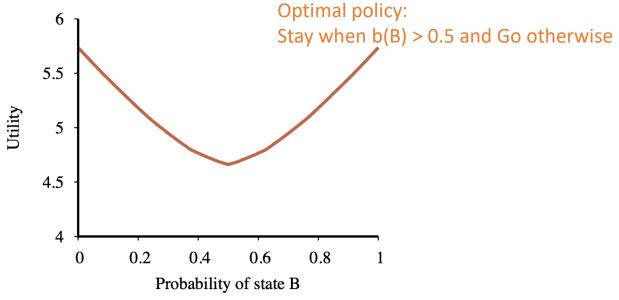
Source: RN Figure 16.15 (b) and (c)

## Value Iteration in POMDP



- In general:
  - Let p be a depth-d conditional plan with initial action a followed by depth (d-1) subplans  $p \cdot e'$  for percept e'  $\alpha_p(s) = \sum_i P(s'|s,a) \left| R(s,a,s') + \gamma \sum_i P(e'|s') \alpha_{p \cdot e'}(s') \right|$
- Given utility function:
  - Executable policy extracted by:
    - looking at which hyperplane is optimal at any given belief state b
    - executing the first action of the corresponding plan
- POMDP Value Iteration (VI):
  - maintains a collection of undominated plans  $\{p\}$  with their utility hyperplanes  $\{alpha\ vectors\ \alpha_p\}$
  - Cf MDP VI computes one utility number, U(s) for each state s

## Example: Utility of Belief-State b(B): d=8



Utility function for optimal 8-step plans

Source: RN Figure 17.15 (d)

# POMDP Value Iteration Algorithm

```
function POMDP-VALUE-ITERATION(pomdp, \epsilon) returns a utility function
  inputs: pomdp, a POMDP with states S, actions A(s), transition model P(s'|s,a),
               sensor model P(e|s), rewards R(s), discount \gamma
           \epsilon, the maximum error allowed in the utility of any state
   local variables: U, U', sets of plans p with associated utility vectors \alpha_p
   U' \leftarrow a set containing just the empty plan [], with \alpha_{[]}(s) = R(s)
   repeat
      U \leftarrow U'
       U' \leftarrow the set of all plans consisting of an action and, for each possible next percept,
           a plan in U with utility vectors computed according to Equation (17.18)
       U' \leftarrow \text{REMOVE-DOMINATED-PLANS}(U')
   until MAX-DIFFERENCE(U, U') \leq \epsilon (1 - \gamma)/\gamma
   return U
                                                                           Source: RN Figure 17.16
```

### Notes on Value Iteration

- Algorithm's complexity depends primarily on how many plans get generated.
  - Given |A| actions and |E| observations, there are  $|A|^{O(|E|^{d-1})}$  distinct depth-d plans.
  - $|A|n^{|E|}$  conditional plans generated at level d+1 before elimination, where n is the no. of conditional plans at level d
  - Elimination of dominated plans to reduce doubly exponential growth
  - P-SPACE hard very inefficient in practice
  - Approximate methods necessary
- Intermediate belief states have lower value than state A and state B
  - In the intermediate states the agent lacks information needed to choose a good action.
  - Information has value and optimal policies in POMDPs often include informationgathering actions.

## Online Methods

Approximate solutions

# Scaling POMDP solvers

- POMDP solvers need to solve two problems
  - Belief tracking/filtering
    - Given the history observed, what is the current belief?
  - Planning
    - Given the current belief, what is the optimal action to take?
- To scale up, need to scale up for both problems

# Online Agent for POMDP

#### Main ideas:

- Represent transition model and sensor model by a dynamic decision network (DDN)
- Belief tracking Inference in DDN Exact inference is computationally intractable
  - No known polynomial time algorithm, as number of state variables grow
- Approximate solution: Deploy a filtering algorithm (exact or particle filtering) to incorporate each new percept and action and to update belief state representation
- Projecting forward possible action sequences and choosing the best one

#### Note:

 A DDN can actually be used as inputs for any POMDP algorithm, include those for value iteration and policy iteration

# Dynamic Decision Network (DDN)

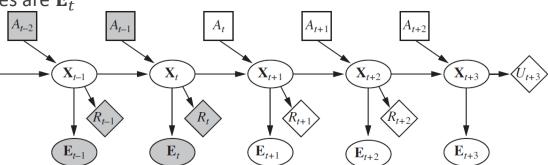
- Execution of POMDP over time can be represented as a DDN
  - Transition and sensor models represented by a Dynamic Bayesian Network (DBN)
  - Add decision and utility nodes to get DDN

#### • In DDN:

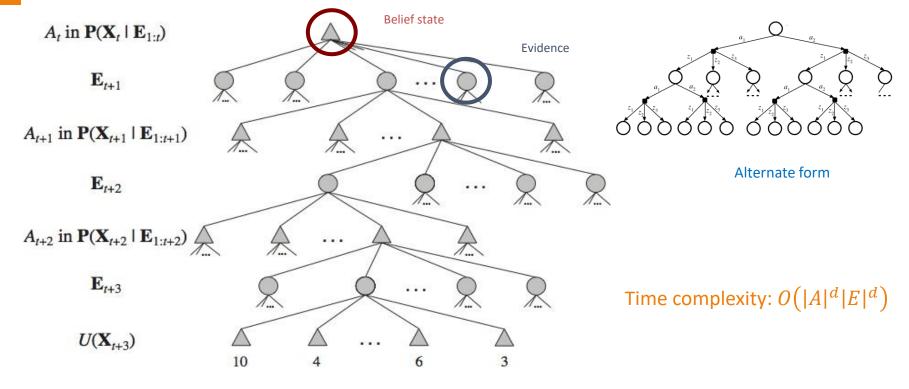
- State  $S_t$  becomes set of variables  $\mathbf{X}_t$
- Evidence/observation variables are  $\mathbf{E}_t$
- Action at time t is  $A_t$
- Transition:  $P(\mathbf{X}_{t+1}|\mathbf{X}_t, A_t)$
- Sensor model:  $P(\mathbf{E}_t|\mathbf{X}_t)$

#### Note:

- Variables with known values are shaded
- Current time is t and agent must decide what to do



## Look-Ahead Solution of POMDP



Source: RN 3e Figure 16.11

## Notes on Online Solution

#### Main ideas:

- Project action sequences forward from current belief state
- Determine utility estimates at projected depth based on future rewards
- Extract decision by backing up utilities from the leaves
  - Averaging chance nodes and maximizing/minimizing decision nodes
- Non-leaf states may have rewards
- Decision nodes correspond to belief states rather than actual states

#### Complexity

• Time complexity of exhaustive search to depth d is  $O(|A|^d|E|^d)$  where |A| is the no. of available actions and |E| is the number of possible percepts

## POMCP<sup>1</sup>

Partially Observable
Monte Carlo Planning

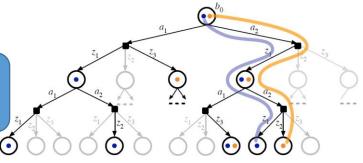
#### MCTS:

- Select
- Expand
- Simulate
- Backup
- To run UCT on POMDP, we need to represent beliefs in the nodes
  - Inference to construct beliefs is intractable in general
  - For MDP, states are easy to generate
  - Beliefs are difficult to generate, so POMCP uses action-observation history
  - History is equivalent to belief assuming same initial belief
- Instead of propagating beliefs forward in a trial
  - POMCP samples a state at the root from the initial belief.
  - Run the simulation using the state to generate action-observation history
  - Only need to sample from P(s'|s,a) then P(e'|s') to generate observation e' for the action-observation history
  - Avoid constructing beliefs

Silver, D. and J. Veness, Monte-Carlo planning in large POMDPs, in Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. 2010: Vancouver, British Columbia, Canada. p. 2164-2172



Determinized Sparse
Partially Observable Tree



Construct search tree differently! (make tree smaller)

Source: Ye et al 2017

- Construct k different search trees
- Each search tree, sample a state at the root to initialize
- At every action node, try all actions on the (single) state at that node
- At every observation node, sample a single observation from the (single) state at node
- Size of the tree:
  - Each of the k trees have size  $A^d$ . Combine together to get tree of size  $O(|A|^d k)$
  - Exponentially smaller than  $|A|^d |E|^d$
- Can show that searching sampled tree is sufficient if a small good policy exists
  - Use heuristics to do anytime search of this tree

<sup>2</sup>Ye, N., et al., *DESPOT: online POMDP planning with regularization*. J. Artif. Int. Res., 2017. **58**(1): p. 231–266.

# POMDP Applications

- Autonomous driving golf cart in UTown
  - https://youtu.be/y\_9VMD\_sQhw
  - DESPOT is used there
  - Works in real-time
- Robotics
- Airborne Collision Avoidance System X
  - ACAS X
  - POMDPs with neural networks for function approximation
- Education
- Dialog system/Chatbot
  - Assistive agent for dementia patients



ADTICLE

Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process



# Summary

#### POMDPs

- Compute optimal policy in partially observable, uncertain domains.
- For finite horizon problems, resulting optimal utility (value) functions are continuous, piecewise linear, and convex.
- In each iteration, no. of linear constraints grows exponentially.
- Approximate solvers can scale to to large size problems; solvers mainly remain in research settings
  - For example, the SARSOP solver from NUS does the Bellman update only on a small subset of beliefs and use heuristics to explore the belief space
  - http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/
- Online search tends to scale better.

## Homework

#### Readings

- RN 16.4, 16.5
- RN 14.2.1, 14.5.3 (Filtering and Particle Filtering Optional)

#### References

- Kaelbling, L.P., M.L. Littman, and A.R. Cassandra, *Planning and acting in partially observable stochastic domains*. Artif. Intell., 1998. **101**(1–2): p. 99–134.
- Silver, D. and J. Veness, Monte-Carlo planning in large POMDPs, in Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. . 2010: Vancouver, British Columbia, Canada. p. 2164-2172.
- Ye, N., et al., *DESPOT: online POMDP planning with regularization.* J. Artif. Int. Res., 2017. **58**(1): p. 231–266.