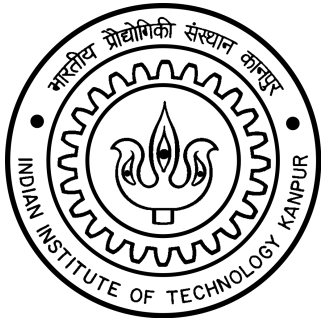


Partially Observable Markov Decision Process (POMDP)

Dr. Gaurav Pandey

Electrical Engineering

IIT Kanpur

Announcement

- HW5: Complete the course reaction survey. You have to take print screen of the page where it shows that you have completed the survey and upload on moodle.

Markov Decision Process (MDP)

- MDP is a fundamental framework used in probabilistic planning. It is commonly used in other areas of science and engineering as well, e.g. game theory, operations research, reinforcement learning, economics, control theory etc.

- Given a sequential decision problem in a fully observable, stochastic environment with a known Markovian state transition model, the MDP is defined by the following components
  - Set of states: S
  - Set of actions: A
  - Initial state: S_0
  - Transition model: T(s, a, s')
  - Reward function: R(s)

MDP solution

- The solution to a MDP is a set of actions that the robot should take in each state to reach the goal. It is also called **policy.**

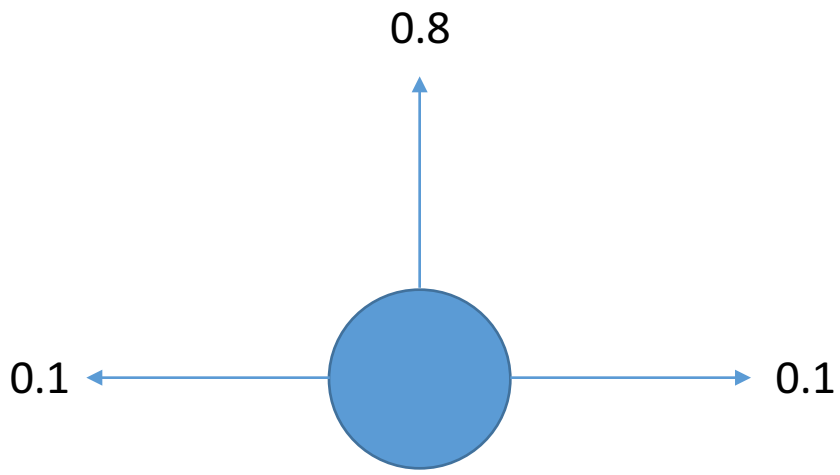
- A policy is a mapping from states to action space. In each state a policy tells the robot what to do next.

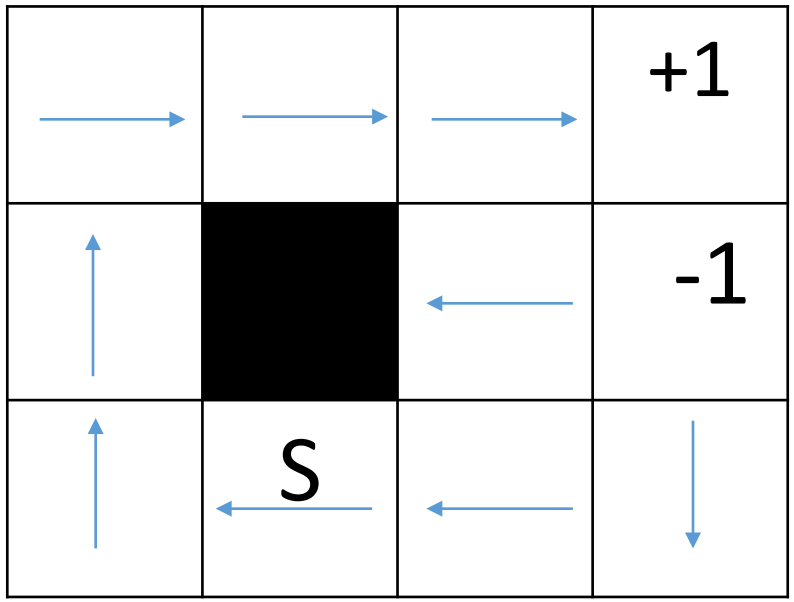
$$\pi = S \rightarrow A$$

- Let $\pi(s)$ be the action that π specifies for “s”. We want to find the optimal policy π^* that takes the robot to goal with maximum reward or minimum cost.

MDP example

- Let R(s) for the free space be -0.01. This means the penalty to take more time to reach the goal is too small (In other words robot has sufficient battery life)! The robot can afford to stay in the maze given that it does not fall in the ditch (-1).

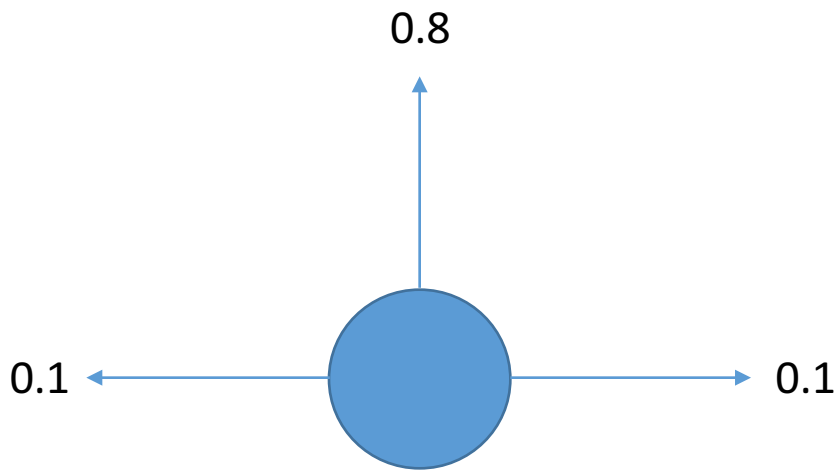


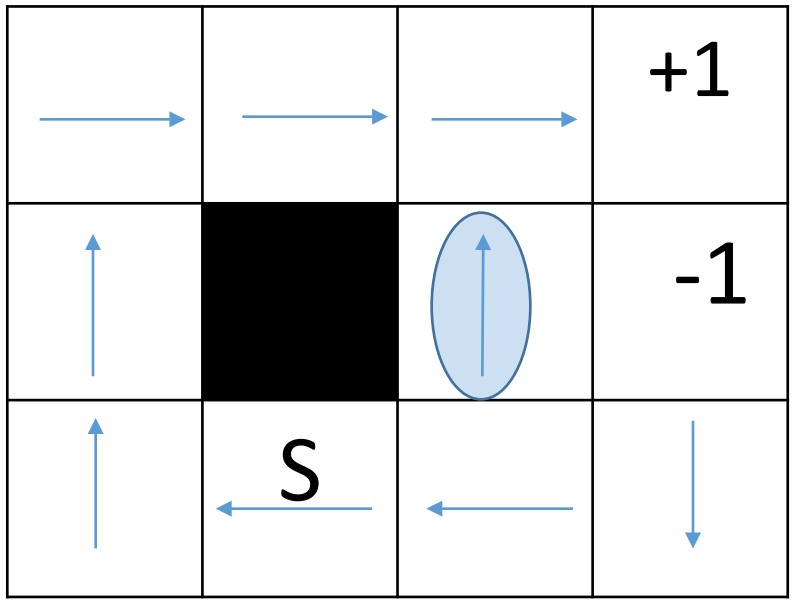


Optimal Policy ?

MDP example

- Let R(s) for the free space be -0.1. This means the penalty to take more time to reach the goal is high now!

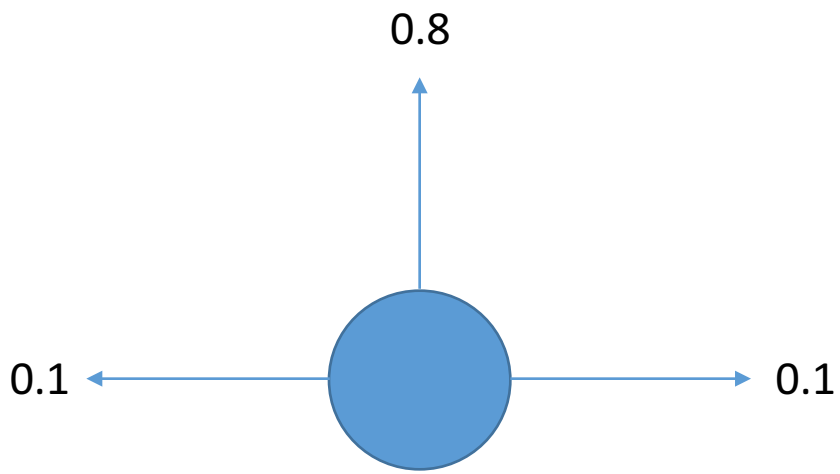


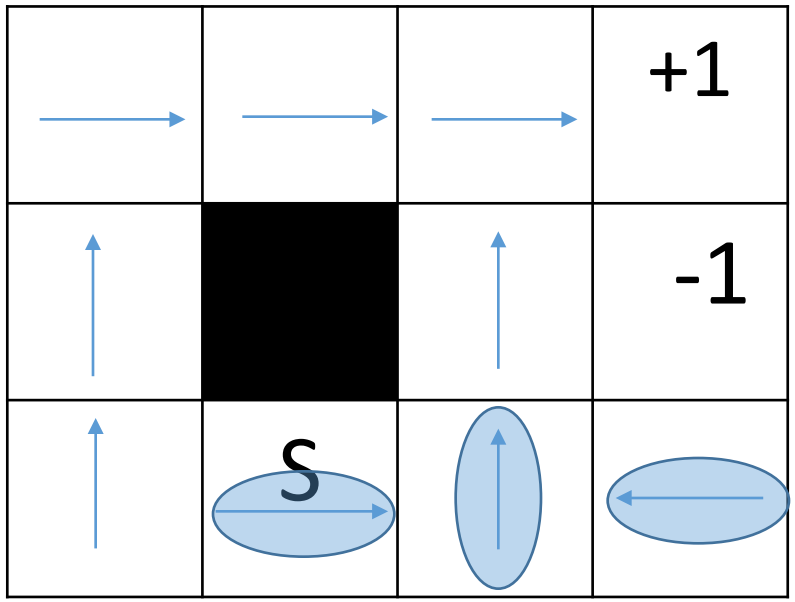


Optimal Policy ?

MDP example

- Let R(s) for the free space be -0.5. This means the penalty to take more time to reach the goal is too high ! The robot should try to reach the goal as quickly as possible.

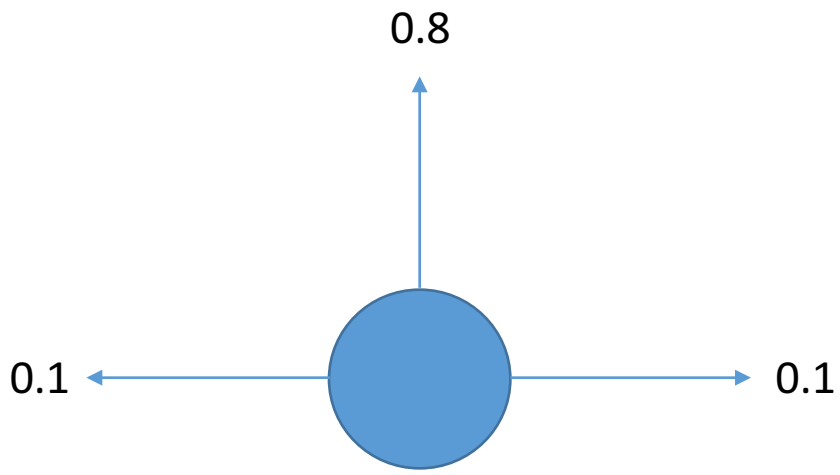


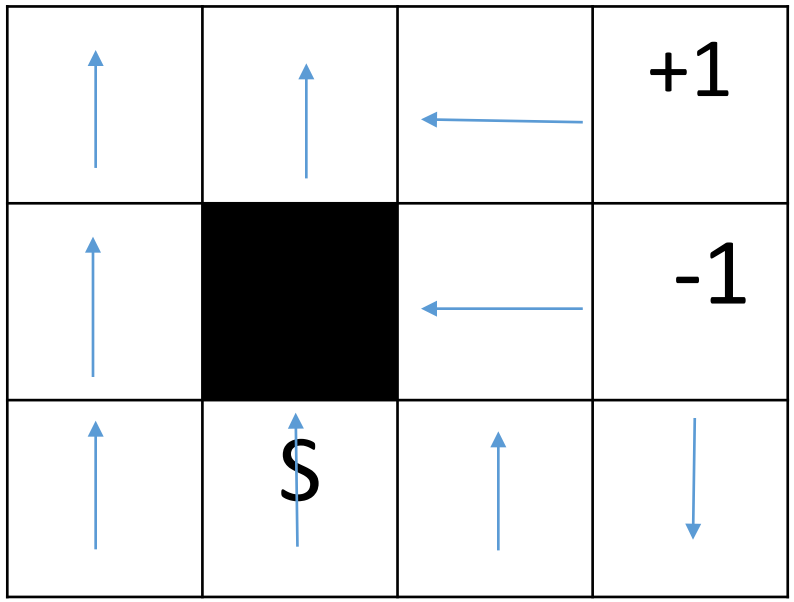


Optimal Policy ?

MDP example

- Let R(s) for the free space be +1.5. This means the robot will try to stay in the maze (not go to the goal at all !)

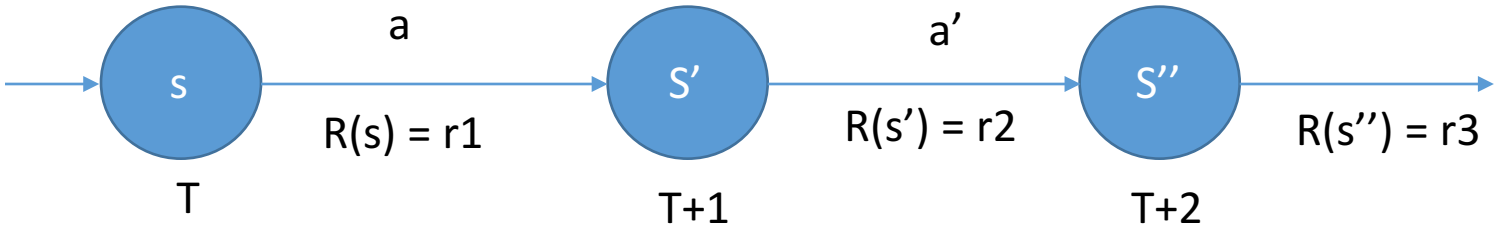




Optimal Policy ?

How to obtain the optimal policy ?

- Note that executing a policy yields a sequence of rewards



- Let R1, R2, …….. be the sequence of random variables representing the reward received by the robot after executing a policy π
- We can define a utility / value function that is a **"quality measure"** of a reward sequence that start in "s" and executes a policy π

Utility / Value function

$$U^{\pi}(s) = E[\sum_{t=0}^{\infty} R(s_t)|\pi, s_0 = s]$$

- This is a measure of how much utility does a state has. If we know the utility of all the states then we can choose an action that takes us to a state that has higher utility!

Expected Linear Additive Utility (ELAU)

- Utility of a state can be expressed as a function of its next state s’

$$
\begin{aligned}
U^{\pi}(s) &= E\left[\sum_{t=0}^{\infty} R(s_t) \mid \pi, s_0 = s\right] \\
&= E\left[R(s_0) + R(s_1) + R(s_2) + \ldots \mid \pi, s_0 = s\right] \\
&= E\left[R(s_0) \mid s_0 = s\right] + E\left[R(s_1) + R(s_2) + \ldots \mid \pi\right] \\
&= R(s) + E\left[\sum_{t=0}^{\infty} R(s_t) \mid \pi, s_0 = s'\right] \\
&= R(s) + U^{\pi}(s')
\end{aligned}
$$

Optimal Policy

- The optimal policy π^* for a given state ‘s’ corresponds to the action ‘a’ that maximizes the expected utility of ‘s’

$$\pi^*(s) = \arg\max_a E[U^\pi(s)]$$

Optimal Policy

$$\pi^*(s) = \arg\max_a E[U^\pi(s)]$$

$$= \arg\max_a E[R(s) + U^\pi(s')]$$

$$= \arg\max_a E[R(s)] + E[U^\pi(s')]$$

$$= \arg\max_a E[U^\pi(s')]$$

$$= \arg\max_a \sum_{s'} T(s, a, s')U(s')$$

- If we know the utility function of a state U(s'), we compute the optimal policy

How to compute the utility function

- Utility of a state U(s) is the expected reward that the robot gets on applying the optimal policy (action) at state ‘s’ ($U^{\pi*}(s) = U(s)$)

$$
\begin{aligned}
U(s) &= \max_a E[U^\pi(s)] \\
&= \max_a E[R(s) + U^\pi(s')] \\
&= R(s) + \max_a E[U^\pi(s')] \\
&= R(s) + \max_a \sum_{s'} T(s, a, s') U(s')
\end{aligned}
$$

- This is an important equation as it relates the utility of a state with the utility of its neighbors !

Utility of a state

- The utility of a state is the immediate reward for that state and the expected utility of the next state, provided the robot choses an optimal action

- For each state we have an equation to compute its utility

$$U(s) = R(s) + \max_a \sum_{s'} T(s, a, s')U(s')$$

- An optimal policy can now be obtained using an iterative approach

Utility of each state

Algorithm 1: Value Iteration

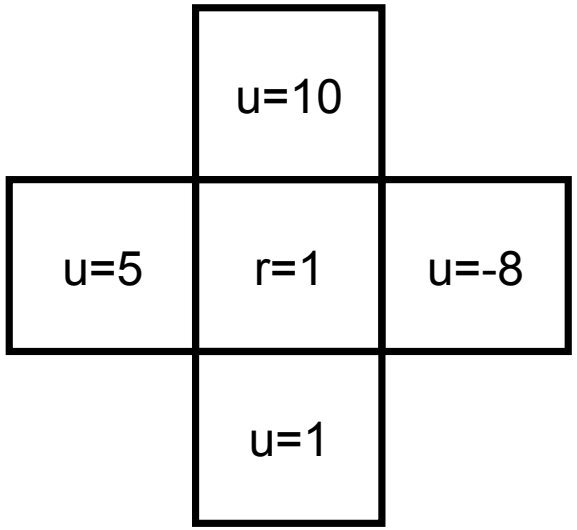
In: An MDP with
– States and action sets S, A,
– Transition model $T(s, a, s')$,
– Reward function $R(s)$,
– Discount factor γ
Out: The utility of all states U

$U' \leftarrow 0$
repeat
  $U \leftarrow U'$
  foreach *state* s *in* S **do**
    $U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s')\, U(s')$
  end
until *close-enough*(U, U')
return U

Slide Credit: Probabilistic Robotics

Value Iteration Example

$$U_{i+1}(s) \leftarrow R(s) + \max_a \sum_{s'} T(s, a, s')\, U_i(s')$$



$$\begin{aligned}
&= reward + \max\{ \\
&\quad 0.1 \cdot 1 + 0.8 \cdot 5 + 0.1 \cdot 10 \quad (\leftarrow), \\
&\quad 0.1 \cdot 5 + 0.8 \cdot 10 + 0.1 \cdot -8 \quad (\uparrow), \\
&\quad 0.1 \cdot 10 + 0.8 \cdot -8 + 0.1 \cdot 1 \quad (\rightarrow), \\
&\quad 0.1 \cdot -8 + 0.8 \cdot 1 + 0.1 \cdot 5 \quad (\downarrow)\} \\
&= 1 + \max\{5.1\,(\leftarrow), 7.7\,(\uparrow), \\
&\quad -5.3\,(\rightarrow), 0.5\,(\downarrow)\} \\
&= 1 + 7.7 \\
&= 8.7
\end{aligned}$$

Slide Credit: Probabilistic Robotics

Discounted Reward

- The utility of a state is often defined as the expected sum of discounted rewards

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

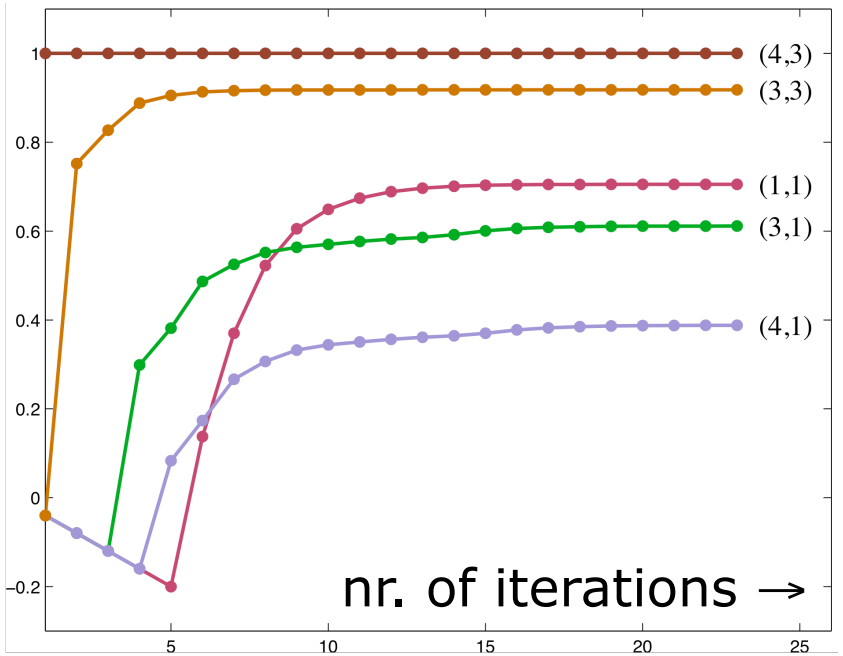
Where gamma is the discount factor which lies between [0,1]

- Discounted reward means that the future rewards are less significant that the current rewards !

Utility of states for the given example

0.812	0.868	0.918	+1
0.762		0.66	-1
0.705	0.655	0.611	0.388

(1,1)



- States far from the goal first accumulates negative utility
- Utilities are higher closer to the goal

Slide Credit: Cyrill Stachniss

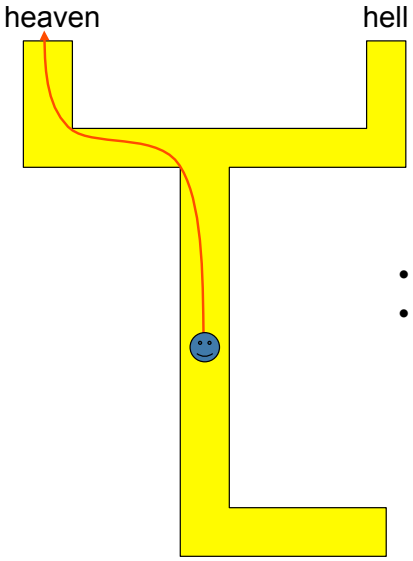
Optimal Policy

- Once we have computed the utility of each state the optimal policy can be computed by

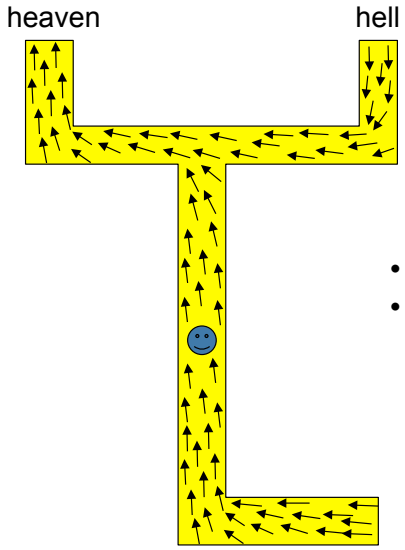
$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s, a, s')\, U(s')$$

- Note that R(s) is the immediate reward of a state and U(s) is the long-term reward of being in a state

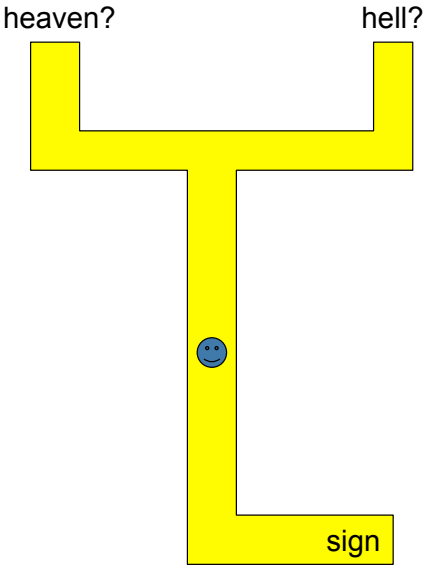
Another Example



- World deterministic
- State observable



- World stochastic
- State observable



Slide Credit: Wolfram Burgard

Partially Observable MDP (POMDP)

- MDP is solved by value iteration over the state space. Assumption is that we **know the state of the robot perfectly** after each action !
- Measurements can be noisy and the state of the robot may not be known exactly. **Instead we only have a belief (Bel(s)) about the robot's current state.**
- **POMDP** is the framework that allows us to compute the optimal policy for the robot when the robot state is only partially observable.
- **POMDP** is solved by value iteration over the **belief space.**

POMDP

- MDP:

$$U(s) = R(s) + \max_a \sum_{s'} T(s, a, s')U(s')$$

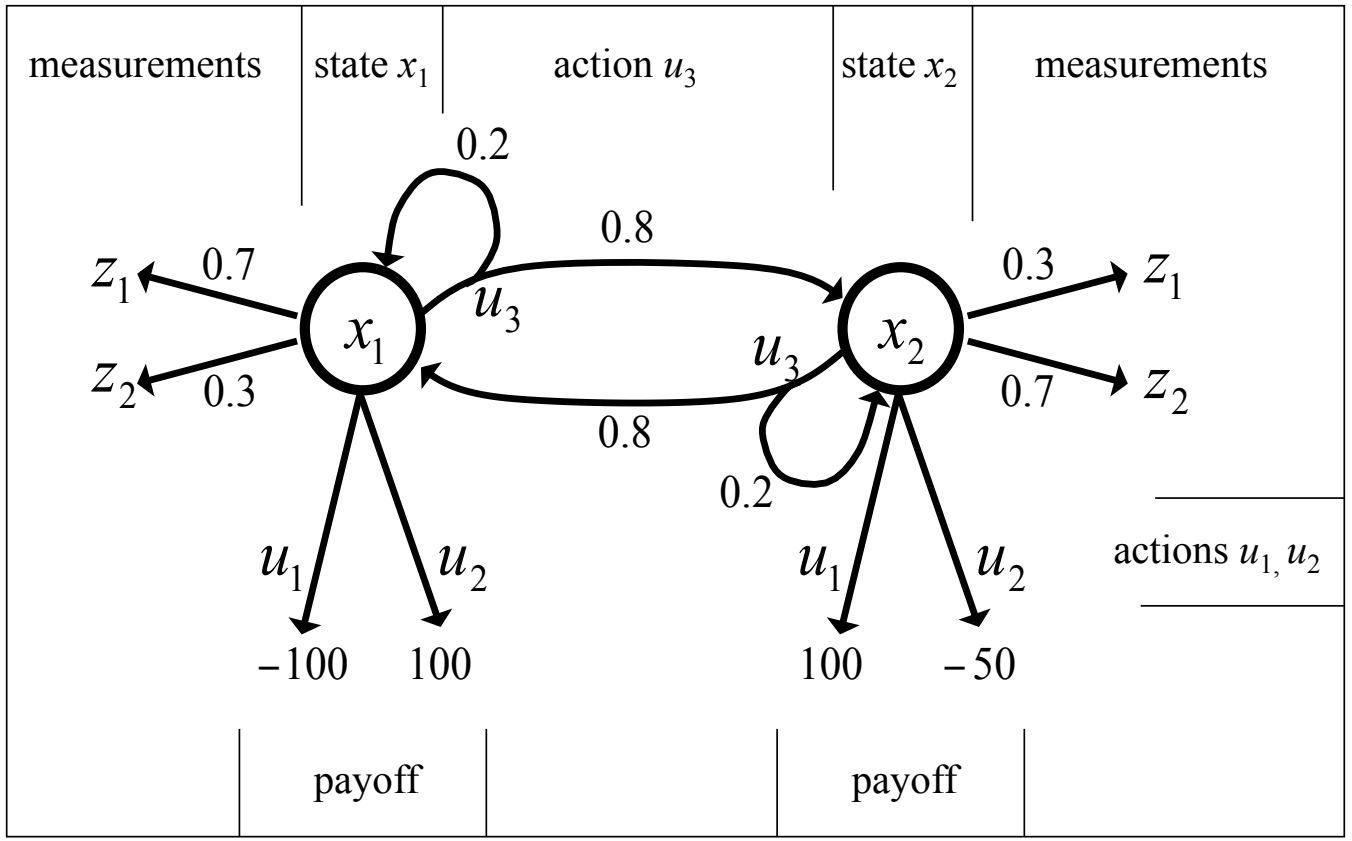
- POMDP:

$$U(Bel(s)) = R(Bel(s)) + \max_a \sum_{s}^{'} T(Bel(s), aBel(s'))U(Bel(s'))$$

- Note: Belief about a state is a probability distribution thus the utility is over an entire probability distribution !

How to compute the utility over belief space?

- Example:



Slide Credit: Probabilistic Robotics

Parameters of the system

- The actions u_1 and u_2 are terminal actions.
- The action u_3 is a sense action that potentially leads to a state transition

$$
\begin{aligned}
r(x_1, u_1) &= -100 & \qquad r(x_2, u_1) &= +100 \\
r(x_1, u_2) &= +100 & \qquad r(x_2, u_2) &= -50 \\
r(x_1, u_3) &= -1 & \qquad r(x_2, u_3) &= -1
\end{aligned}
$$

$$
\begin{aligned}
p(x_1'|x_1, u_3) &= 0.2 & \qquad p(x_2'|x_1, u_3) &= 0.8 \\
p(x_1'|x_2, u_3) &= 0.8 & \qquad p(z_2'|x_2, u_3) &= 0.2
\end{aligned}
$$

$$
\begin{aligned}
p(z_1|x_1) &= 0.7 & \qquad p(z_2|x_1) &= 0.3 \\
p(z_1|x_2) &= 0.3 & \qquad p(z_2|x_2) &= 0.7
\end{aligned}
$$

Slide Credit: Probabilistic Robotics

Expected Reward

- Since we do not know the state, we only know the belief b = {p_1, p_2}, the expected payoff/reward for any action under this belief is given by:

$$\begin{aligned} r(b,u) &= E_x[r(x,u)] \\ &= \int r(x,u)p(x)\ dx \\ &= p_1\ r(x_1,u) + p_2\ r(x_2,u) \end{aligned}$$

- Note that p_1 = Bel(x_1) and p_2 = Bel(x_2)

Slide Credit: Probabilistic Robotics

Expected reward for our example

- If we are totally certain that we are in state x and execute action u then we will get a reward of ?
- Expected reward for belief “b”

$$
\begin{aligned}
r(b, u_1) &= -100\, p_1 + 100\, p_2 \\
&= -100\, p_1 + 100\, (1 - p_1)
\end{aligned}
$$

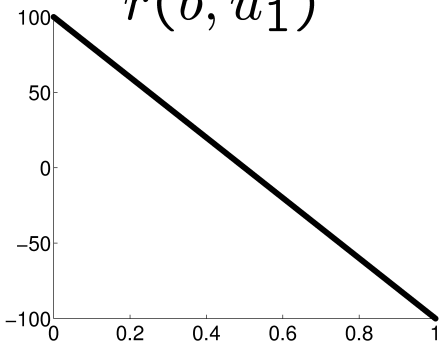
$$
r(b, u_2) = 100\, p_1 - 50\, (1 - p_1)
$$

$$
r(b, u_3) = -1
$$

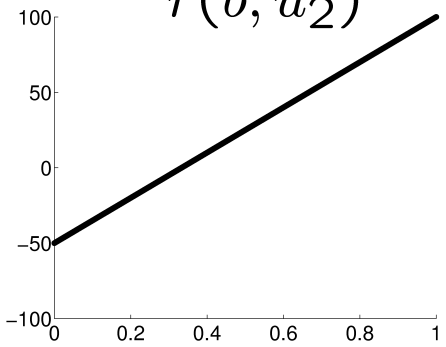
Slide Credit: Probabilistic Robotics

Payoffs in the given example

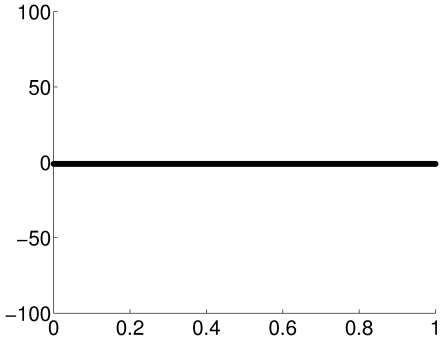
$r(b, u_1)$



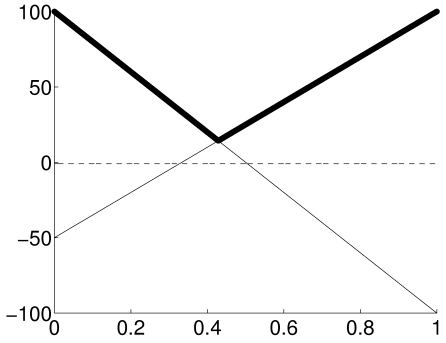
$r(b, u_2)$



$r(b, u_3)$



$V_1(b) = max_u r(b, u)$



Slide Credit: Probabilistic Robotics

Optimal Policy for T=1

- Given that we have a finite POMDP with T=1, we will use $V_1(b)$ to determine the optimal policy:

$$
\begin{aligned}
V_1(b) &= \max_u \; r(b,u) \\
&= \max \left\{ \begin{array}{cc} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ -1 & \end{array} \right\}
\end{aligned}
$$

$$
\pi_1(b) = \begin{cases} u_1 & \text{if } p_1 \leq \frac{3}{7} \\ \\ u_2 & \text{if } p_1 > \frac{3}{7} \end{cases}
$$

Slide Credit: Probabilistic Robotics

Increasing the time horizon to T=2

- In this case the robot can also sense the environment using the sensing action u_3
- Suppose we get an observation z_1

$Bel_2(x_1) = p(x_1 \mid z_1)$

$$V_1(b \mid z_1) = \max \left\{ \begin{array}{ll} -100 \cdot \frac{0.7\, p_1}{p(z_1)} & +100 \cdot \frac{0.3\, (1-p_1)}{p(z_1)} \\ 100 \cdot \frac{0.7\, p_1}{p(z_1)} & -50 \cdot \frac{0.3\, (1-p_1)}{p(z_1)} \end{array} \right\}$$

$$= \frac{1}{p(z_1)} \max \left\{ \begin{array}{ll} -70\, p_1 & +30\, (1-p_1) \\ 70\, p_1 & -15\, (1-p_1) \end{array} \right\}$$

Slide Credit: Probabilistic Robotics

Expected value after measurement

- Since we do not know what measurement we will get, we consider the expected value

$$
\begin{aligned}
\bar{V}_1(b) &= E_z[V_1(b \mid z)] \\
&= \sum_{i=1}^{2} p(z_i)\, V_1(b \mid z_i) \\
&= \max \left\{ \begin{array}{ll} -70\, p_1 & +30\, (1-p_1) \\ 70\, p_1 & -15\, (1-p_1) \end{array} \right\} \\
&\quad + \max \left\{ \begin{array}{ll} -30\, p_1 & +70\, (1-p_1) \\ 30\, p_1 & -35\, (1-p_1) \end{array} \right\}
\end{aligned}
$$

Slide Credit: Probabilistic Robotics

Resulting value function

- The four possible combination give the following value function

$$\bar{V}_1(b) = \max \left\{ \begin{array}{cccc} -70\,p_1 & +30\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ -70\,p_1 & +30\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \end{array} \right\}$$

Pruning

$$= \max \left\{ \begin{array}{cc} -100\,p_1 & +100\,(1-p_1) \\ +40\,p_1 & +55\,(1-p_1) \\ +100\,p_1 & -50\,(1-p_1) \end{array} \right\}$$

Slide Credit: Wolfram Burgard

Account for state transitions

- Since sensing action can potentially change the state, we need to take these state changes into account while calculating the value function. Since we do not know which state we are in at present, we can compute the expected value of the new belief $p(x_1 \mid x, u_3)$

$$
\begin{aligned}
p_1' &= E_x[p(x_1 \mid x, u_3)] \\
&= \sum_{i=1}^{2} p(x_1 \mid x_i, u_3) p_i \\
&= 0.2p_1 + 0.8(1 - p_1) \\
&= 0.8 - 0.6p_1
\end{aligned}
$$

Slide Credit: Probabilistic Robotics

Resulting value function after executing the sensing action

$$\bar{V}_1(b \mid u_3) = \max \left\{ \begin{array}{rr} 60\, p_1 & -60\,(1-p_1) \\ 52\, p_1 & +43\,(1-p_1) \\ -20\, p_1 & +70\,(1-p_1) \end{array} \right\}$$

Slide Credit: Probabilistic Robotics

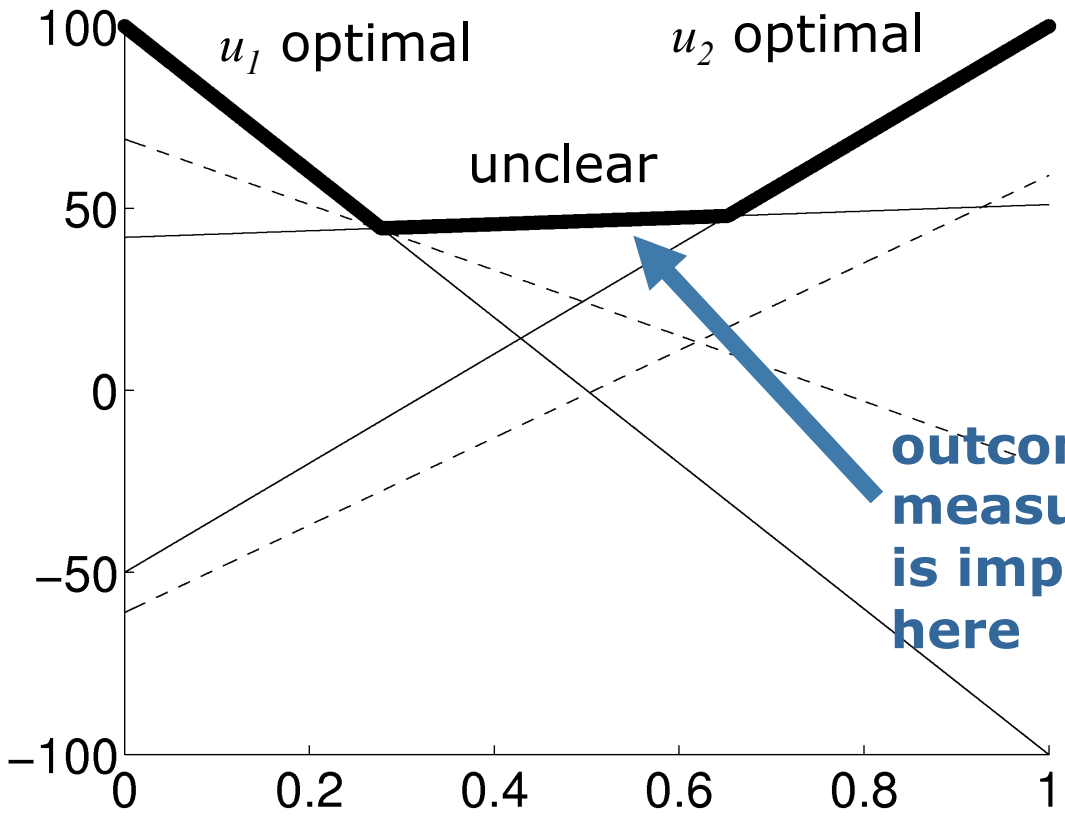
Value function for T=2

- Taking into account that the robot can take action u_1 or u_2 directly, or it can first perform sensing action u_3 then take the terminal actions

$$\bar{V}_2(b) = \max \left\{ \begin{array}{cc} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ 51\, p_1 & +42\,(1-p_1) \end{array} \right\}$$

Slide Credit: Probabilistic Robotics

Value function graphically



Here u_3 is optimal

Slide Credit: Probabilistic Robotics

Summary

- POMDPs compute the optimal action in partially observable, stochastic domains.
- For finite horizon problems, the resulting value functions are piecewise linear and convex.
- In each iteration the number of linear constraints grows exponentially.
- POMDPs so far have only been applied successfully to very small state spaces with small numbers of possible observations and actions.

Slide Credit: Wolfram Burgard