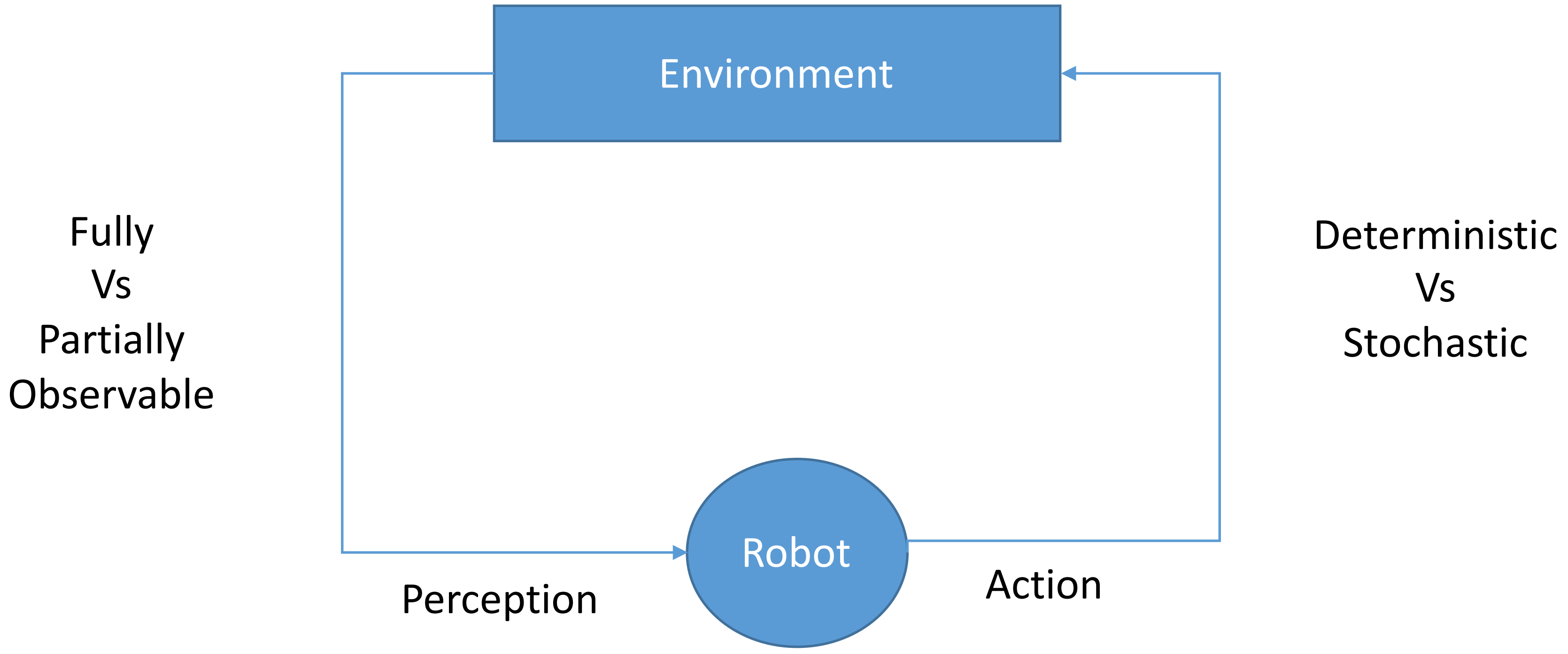
Path Planning
Markov Decision Process (MDP)

Dr. Gaurav Pandey

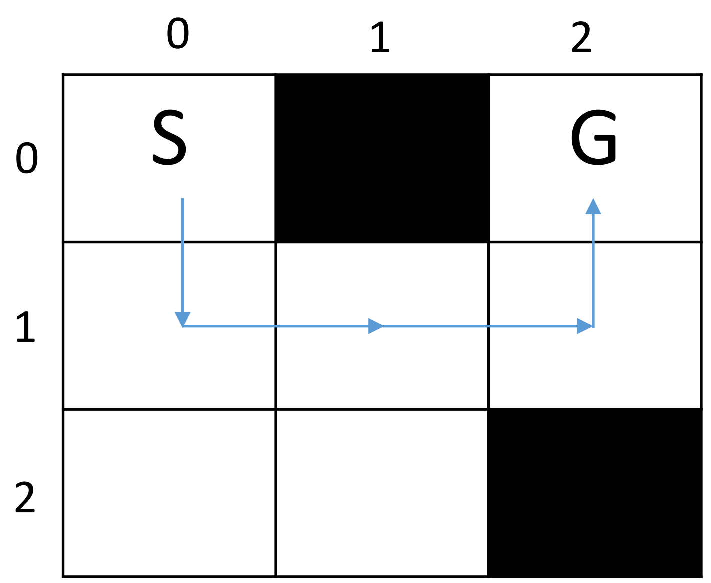
Electrical Engineering

IIT Kanpur

Path Planning



Deterministic path planning

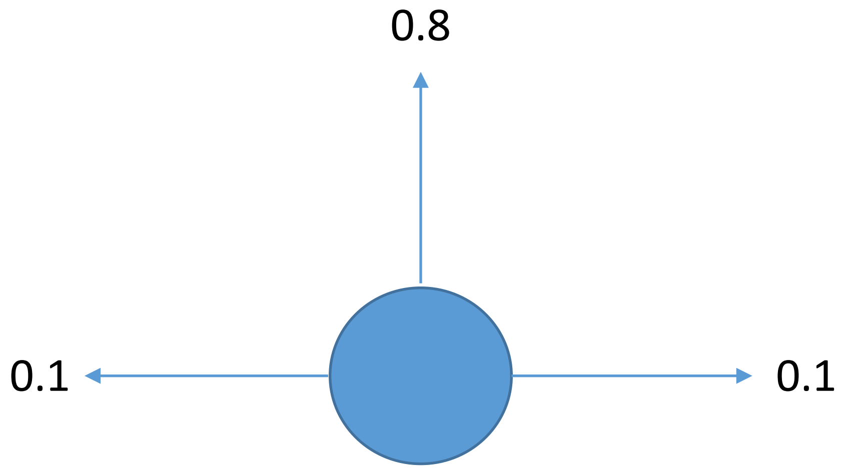


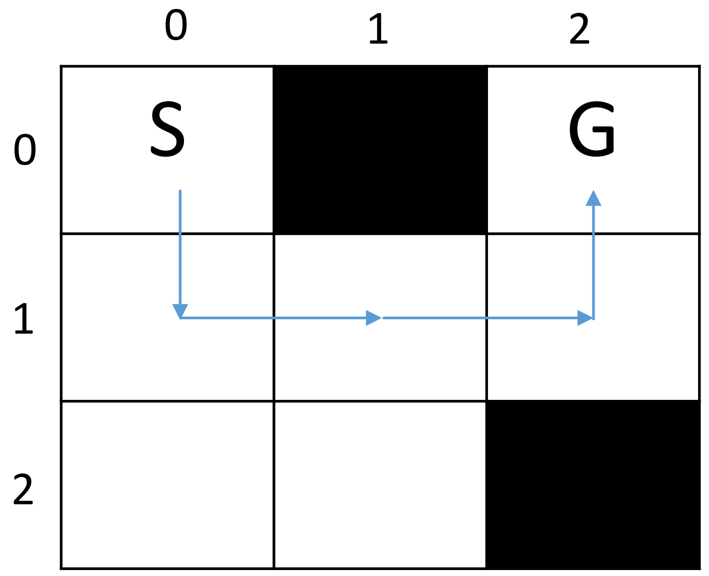
If the state transition is deterministic taking actions DRRU will always take you to the goal. A* or RRT can find this path !

What happens when the transition is non-deterministic !!

Non-deterministic actions

- Non-deterministic state transition





- A given sequence of action may not take you to the same state always!

Non-deterministic actions

- When the actions are non-deterministic then we do not find an action sequence but we find a **policy**, which is action taken by the robot depending upon the current state.

- How can we find this **optimal policy** ?

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: Definitions

- Transition Model T(s,a,s'): This is the probability to reach the next state s' from current state s by choosing action a.

$$T(s, a, s') = P(s' \mid s, a)$$

- This is similar to the motion model. Markov assumption holds. The state transition probability depends only on the current state and not on the history of earlier states.

MDP: Definitions

- Reward R(s): In each state “s” the robot receives a reward R(s).
- The reward may be positive or negative but it must be bounded.
- For instance in the 3x3 grid map
  - Reward in all empty/free cells is -0.01
  - Reward in goal cell = +1
  - Reward in occupied cell = -1
- A negative reward in free cells gives the robot an incentive to reach the goal quickly.

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.**

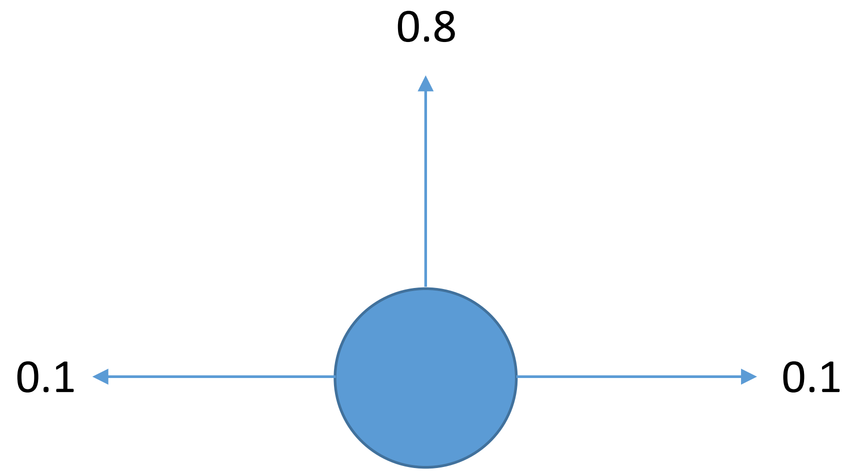
- A policy is a mapping from states to action space. In each state a policy tells the robot what to do next.

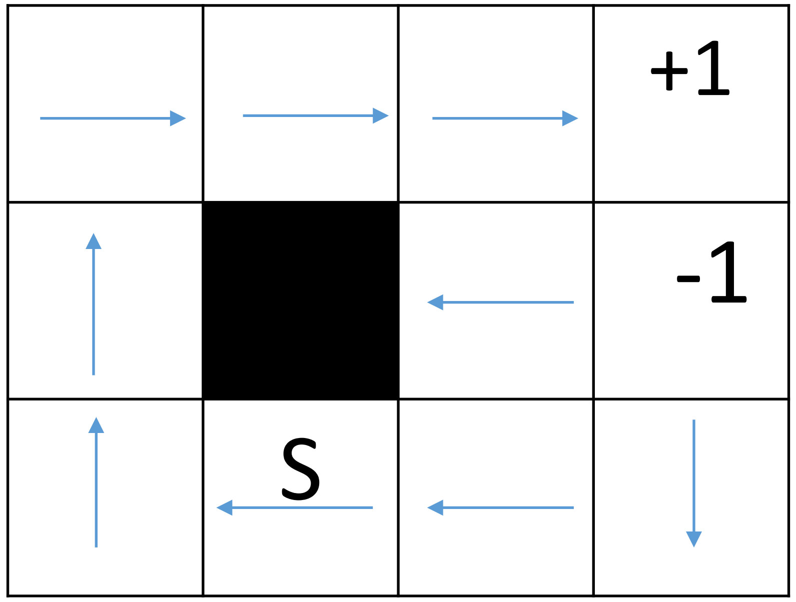
$$\pi = S \rightarrow A$$

- 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).

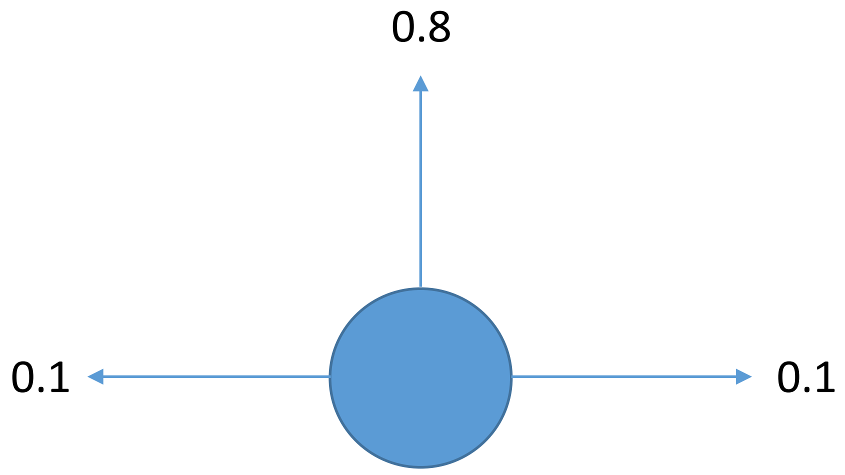


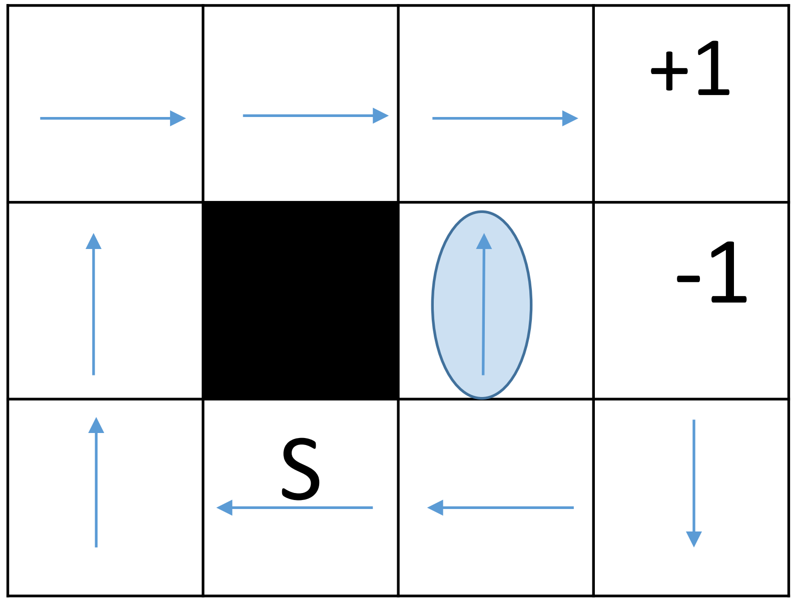


Optimal Policy ?

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!

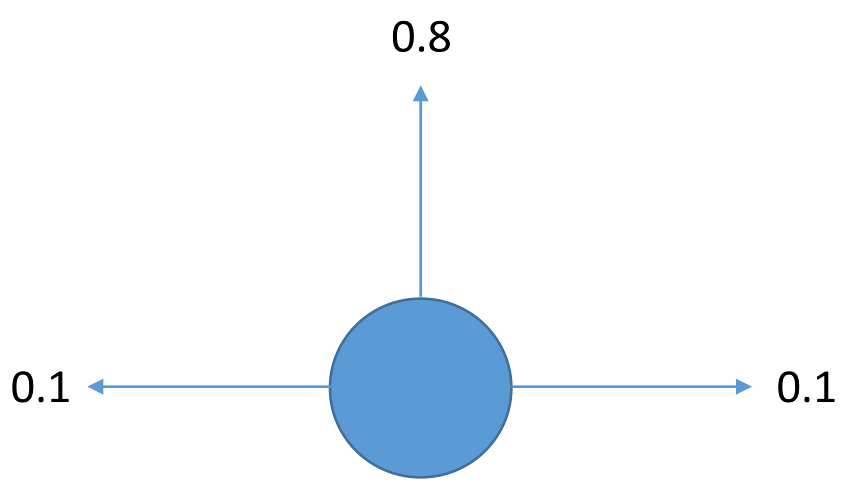


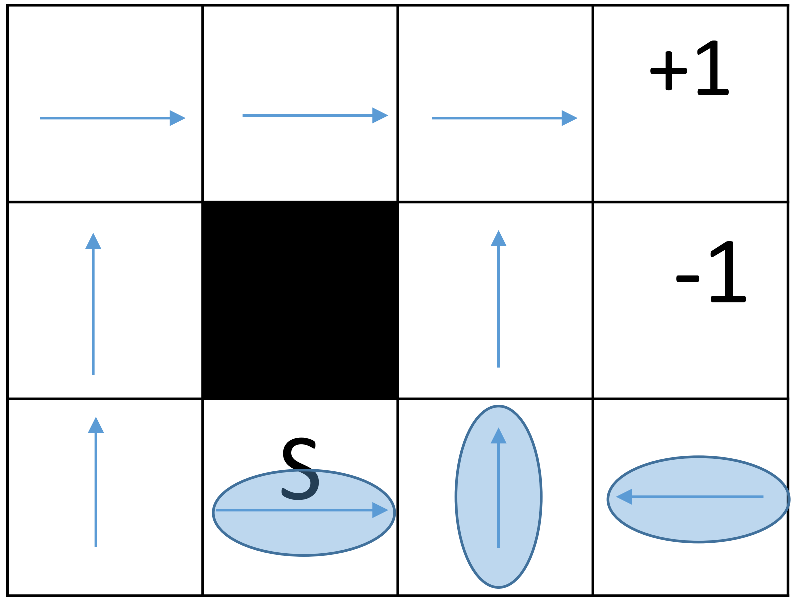


Optimal Policy ?

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.

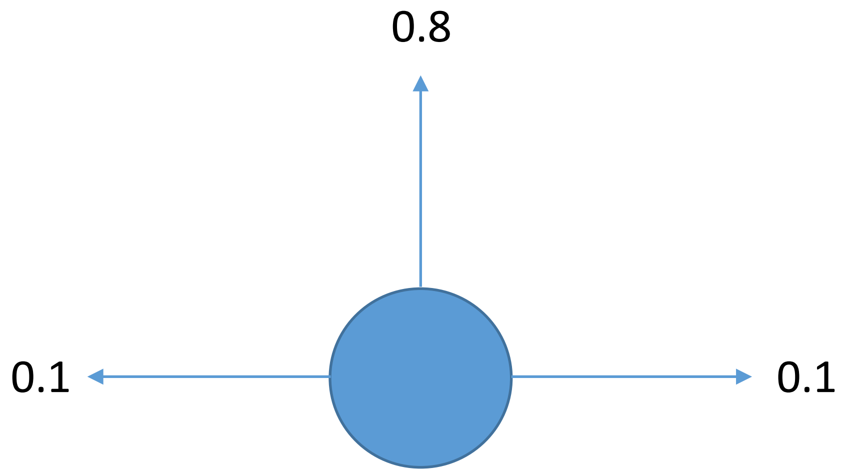


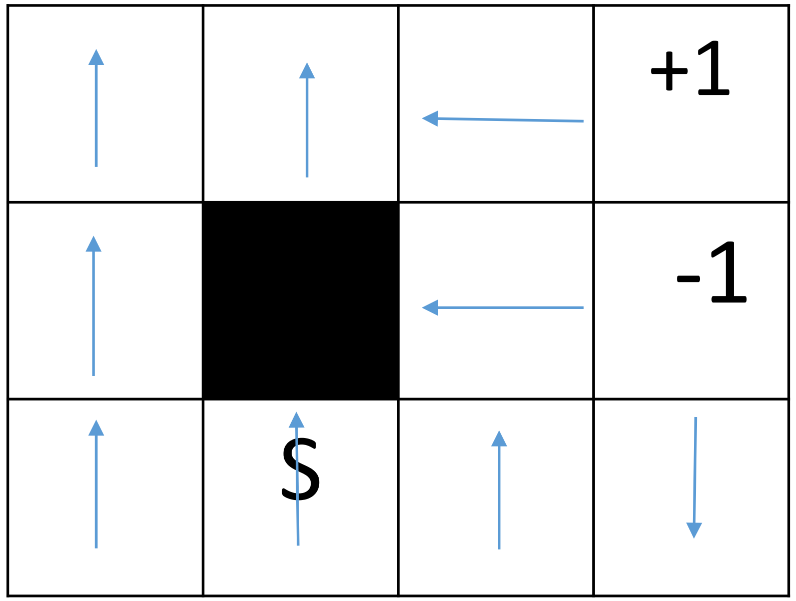


Optimal Policy ?

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 !)

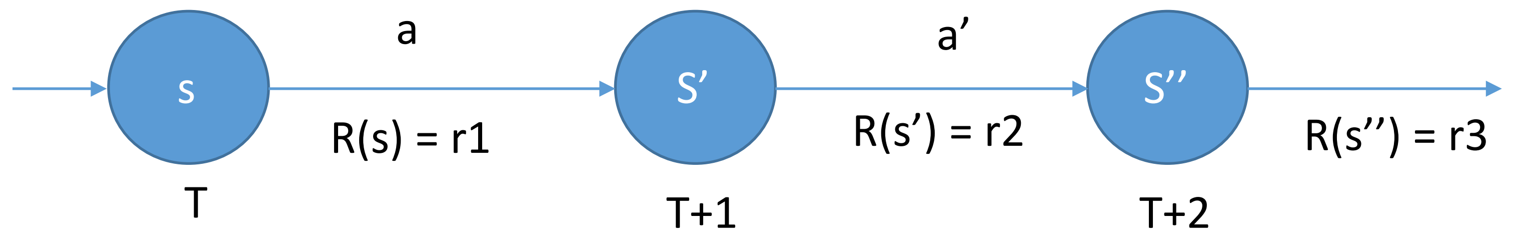




Optimal Policy ?

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

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

- 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

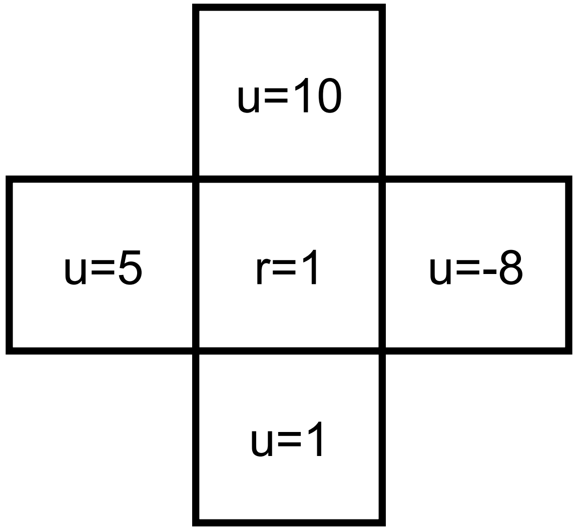
Algorithm 1: Value Iteration

```
In: An MDP with
– States and action sets $S$, $A$,
– Transition model $T(s, a, s')$,
– Reward function $R(s)$,
– Discount factor $\gamma$
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 $U$
```

Value Iteration Example

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



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

Slide Credit: Cyrill Stachniss

Discounted Reward

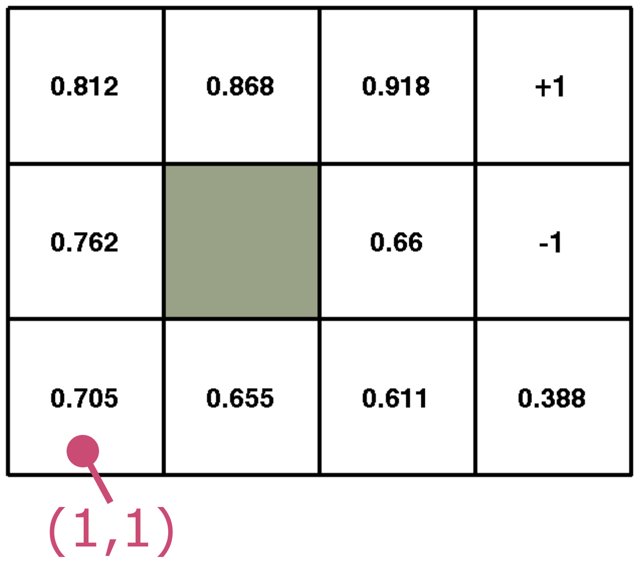
- The utility of a state is often defined as the expected sum of discounted rewards

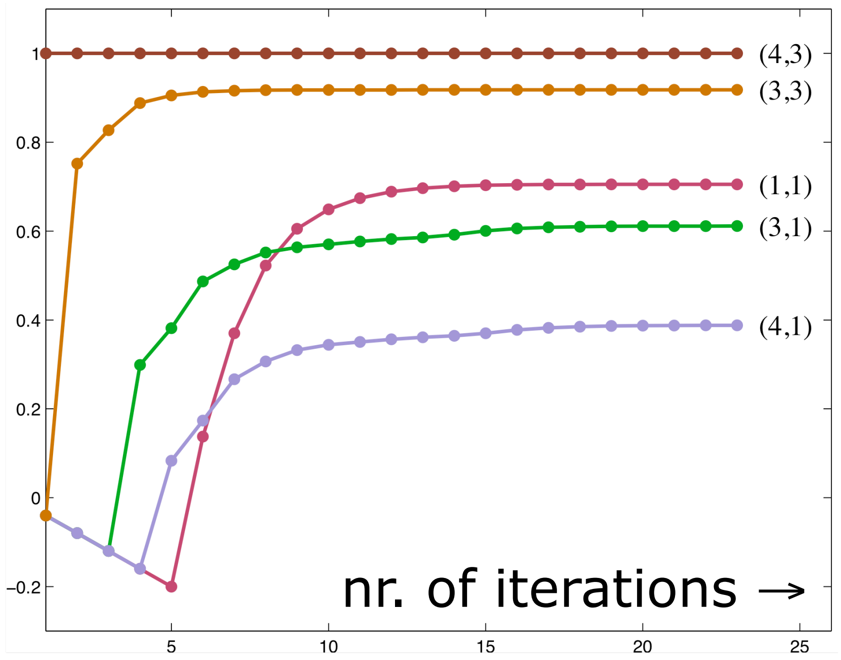
$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \underline{\gamma^{t}} R(s_t) \mid \pi, s_0 = s\right]$$

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





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

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 of Optimal Policy