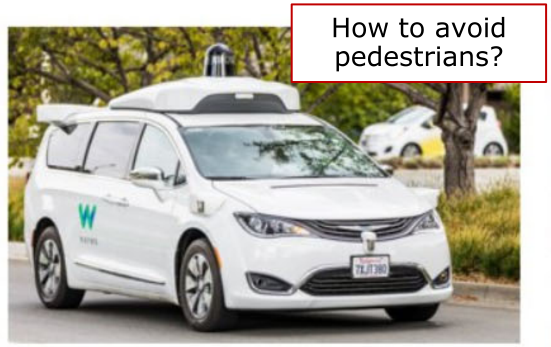
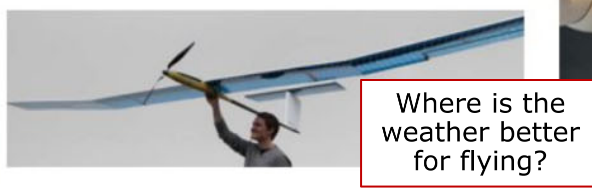
3 – What is Decision-Making?

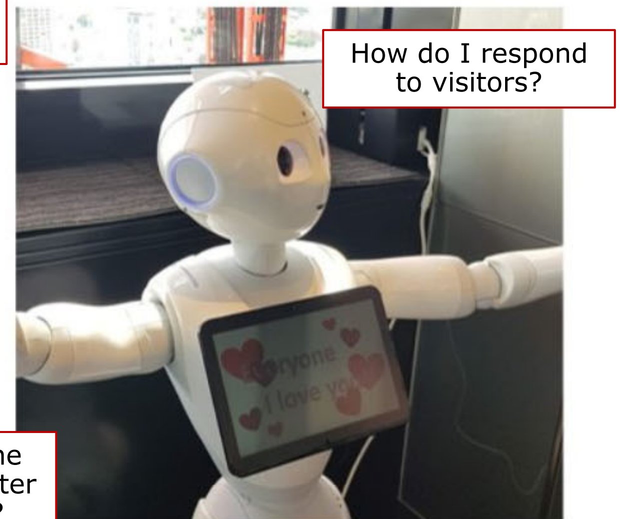
Dr. Marija Popović

UNIVERSITÄT BONN

Motivation



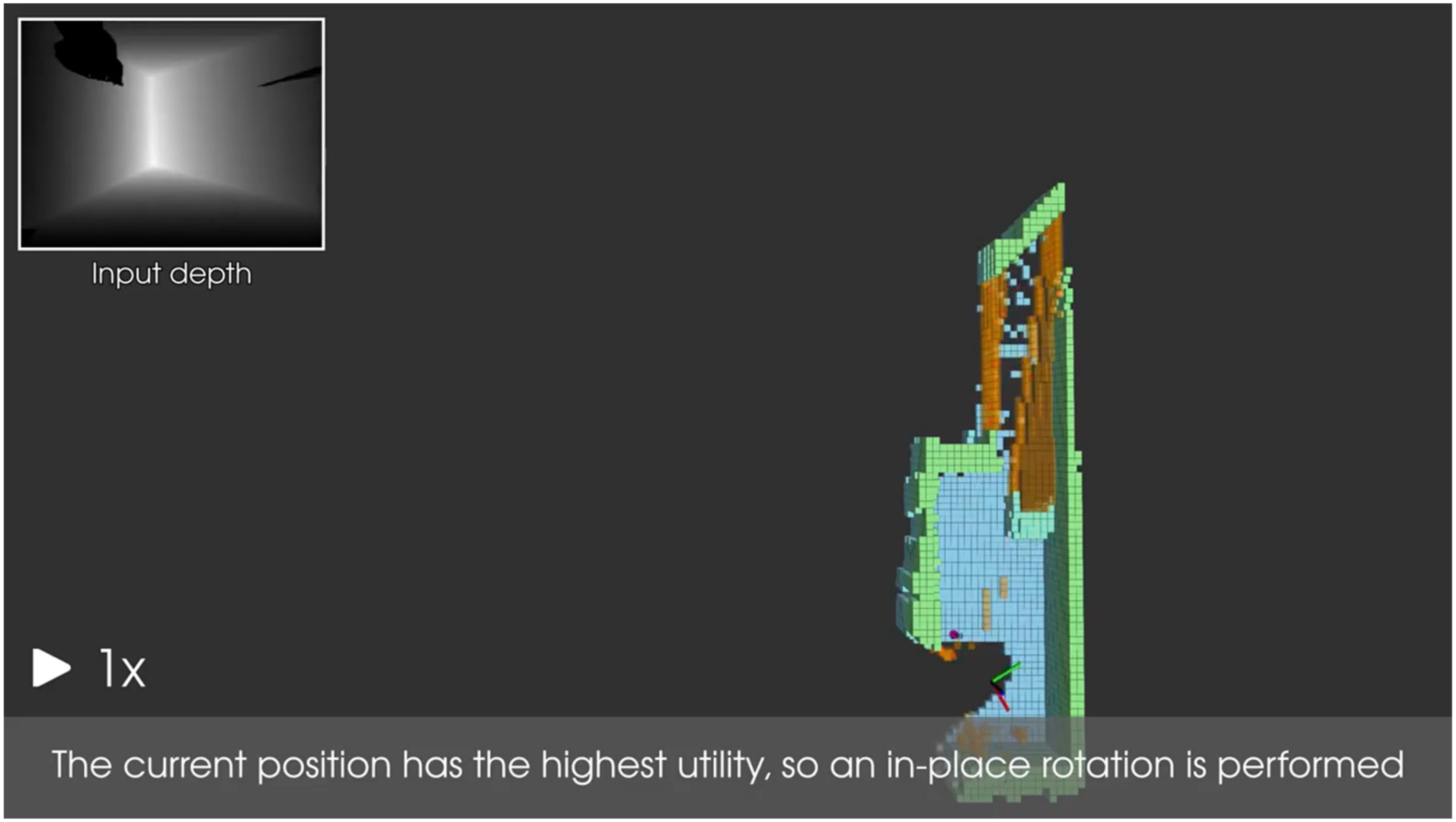




http://www.cs.columbia.edu/~allen/F17/NOTES/Lecture_2.pdf
http://asl.stanford.edu/aa274a/pdfs/lecture/lecture_1.pdf
Hitz et al. (2014), "Fully autonomous focused exploration for robotic environmental monitoring," in: IEEE ICRA.
Lehnert et al. (2020), "Performance improvements of a sweet pepper harvesting robot in protected cropping environments," in: JFR. 37(7): pp.1197-1223.

Motivation

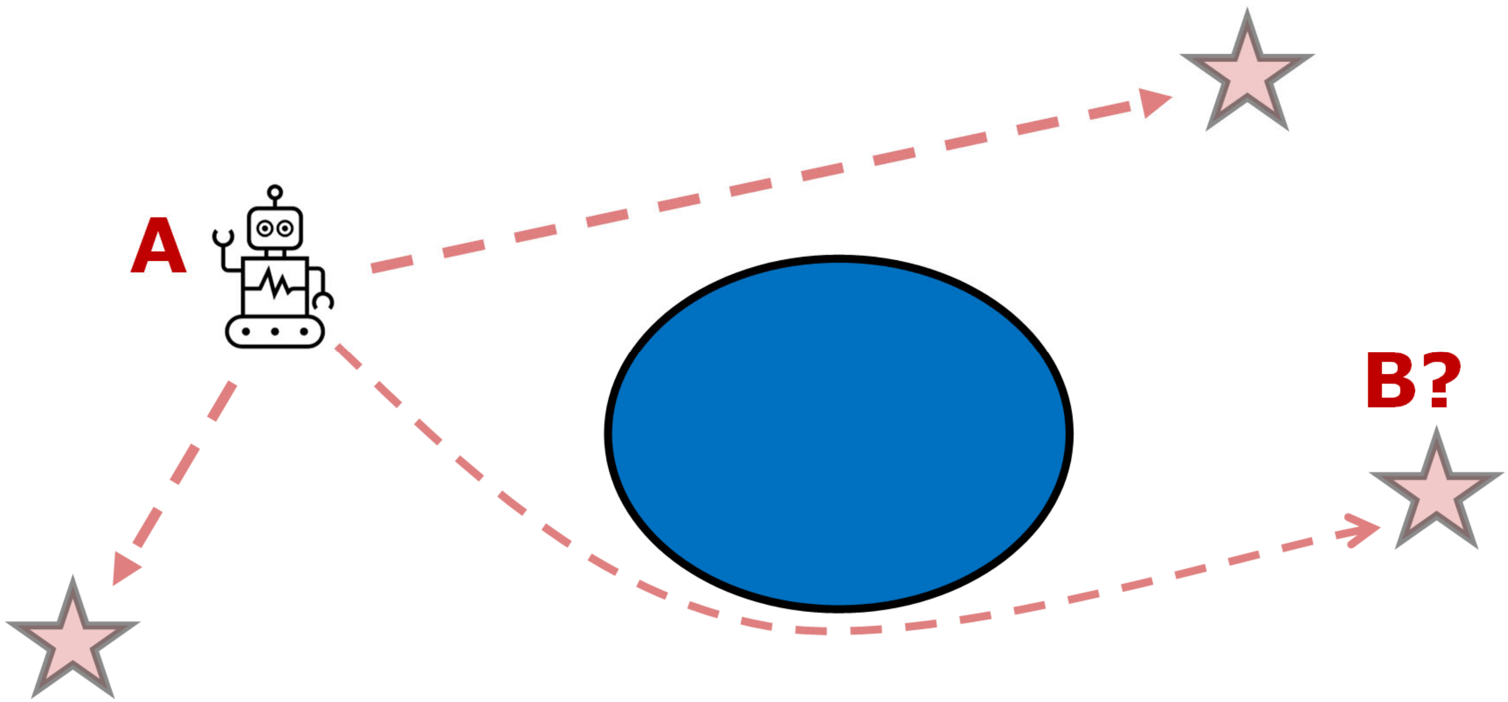
- Robotic exploration



Dai et al. (2020), “Fast Frontier-based Information-driven Autonomous Exploration with an MAV,“ in: IEEE ICRA. pp.9570-9576.

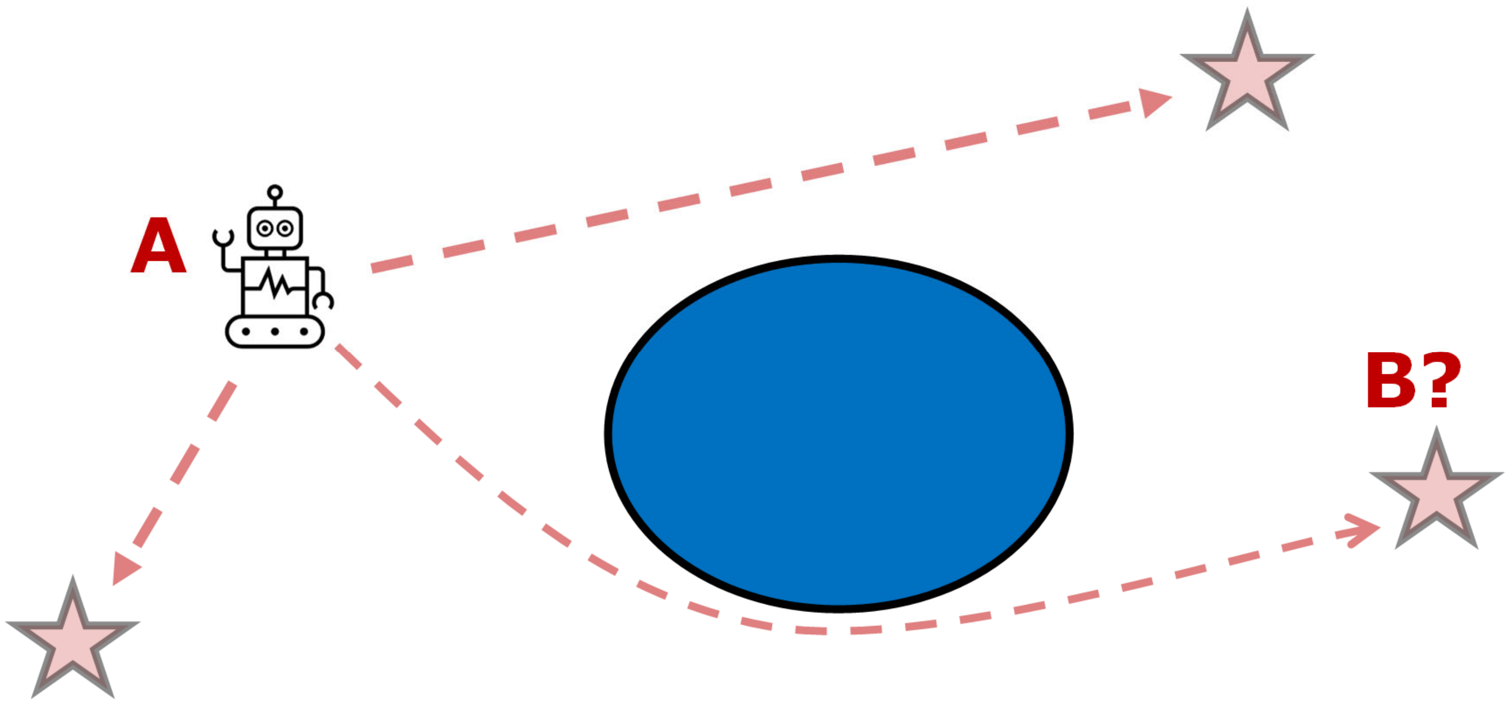
What is Decision-Making?

- Find goal location(s) (point B) to fulfill a particular task – *where*?



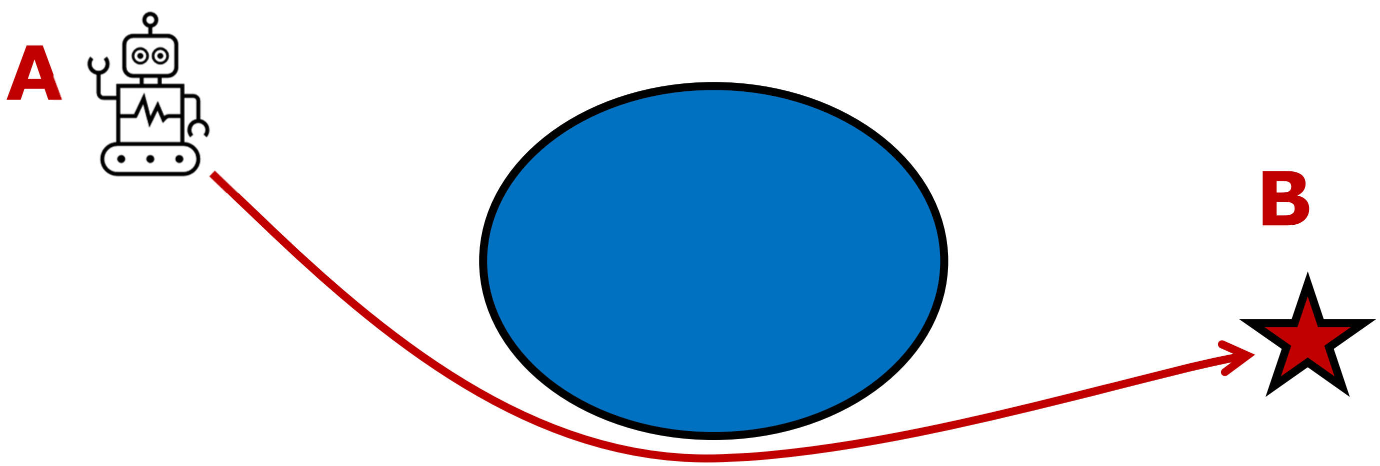
Planning vs. Decision-Making

- **Decision-making**: Where to go?



Planning vs. Decision-Making

- **Decision-making**: Where to go?
- **Planning**: How to get there?

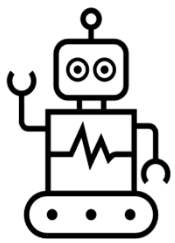


Planning vs. Decision-Making

- **Decision-making**: Where to go?
- **Planning**: How to get there?
- Motion planning is sequential decision-making
- Decision-making aspects:
  - Different objective functions (rewards)
  - Uncertainty (actuation and sensing)
  - Discrete problems

Agent and Environment

- **Agent**: Learner and decision-maker
- **Environment**: Everything outside the agent



- **Action** A: Choice made by the agent
- **Observation** O: Information the agent gets about the environment
- **Reward** R: Scalar feedback signal defining the agent's goal

Reward

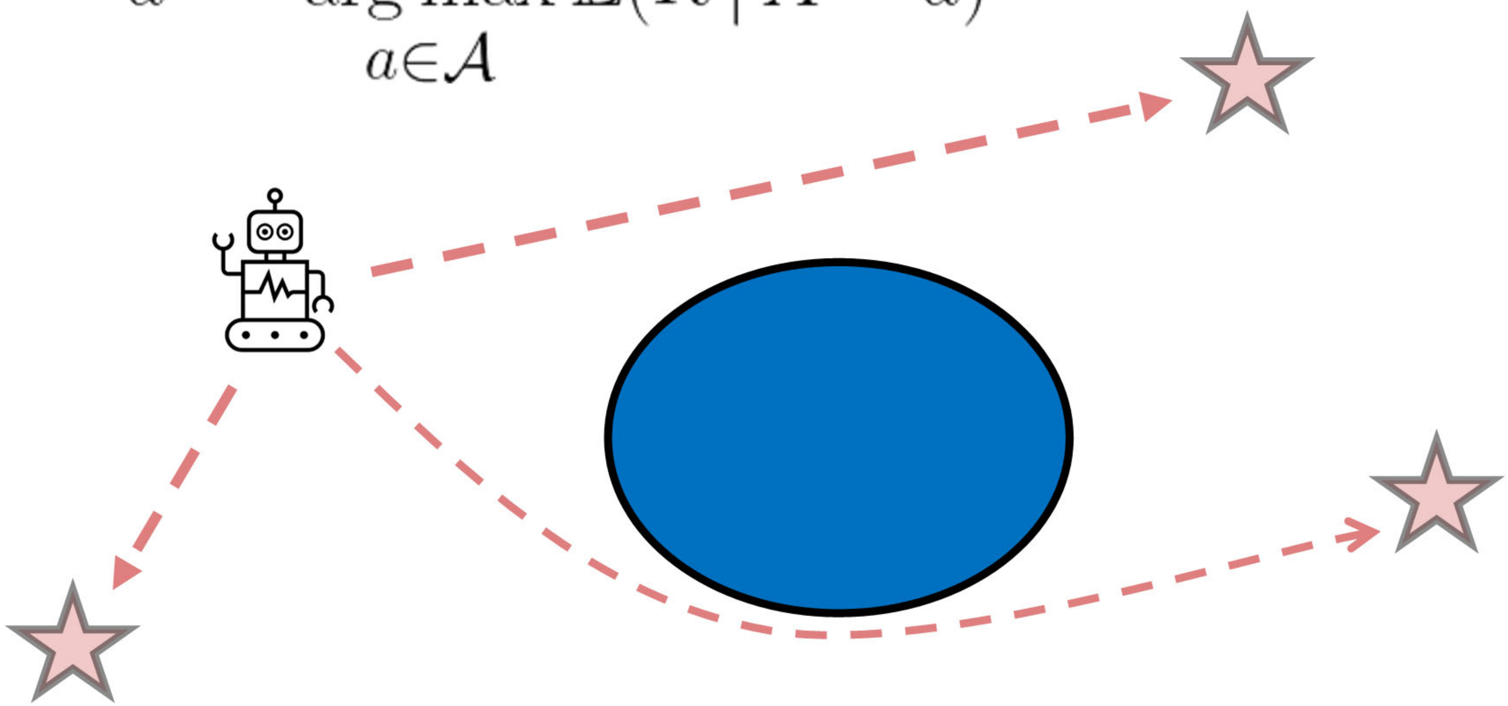
- **Reward** R_t: Scalar feedback signal defining the agent's goal
- How well the agent is doing at step t
- **Goal**: Maximise expected cumulative reward

- Examples:
  - UAV exploring an unknown environment
    - + reward for finding unobserved areas
  - Quadrotor control
    - + reward for stability (position and derivatives)
  - Autonomous surface vehicle monitoring lake
    - + reward for measurements with high bacteria concentration

Single-Stage Decision-Making

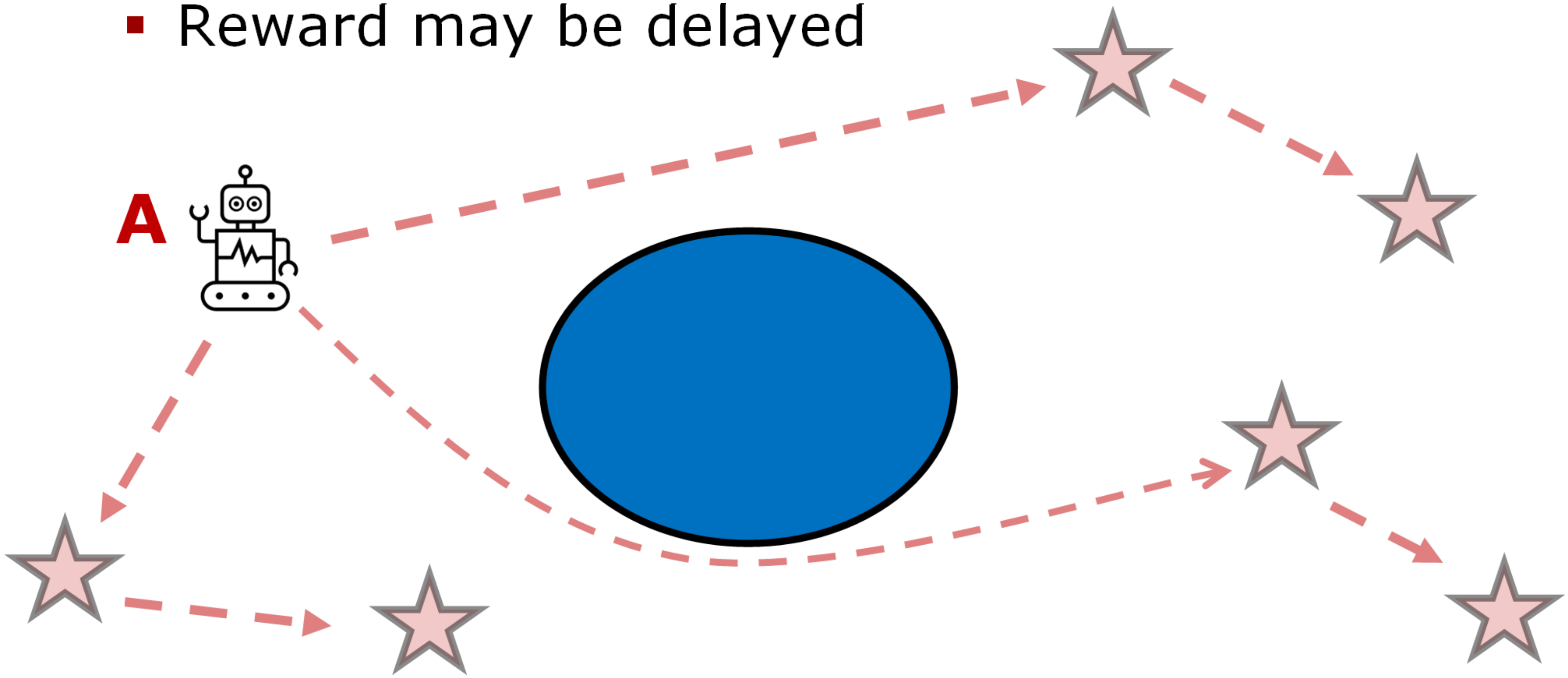
- **Goal**: Choose one action to maximise the expected reward
- Optimal action:

$$a^* = \arg\max_{a \in \mathcal{A}} \mathbb{E}(R \mid A = a)$$



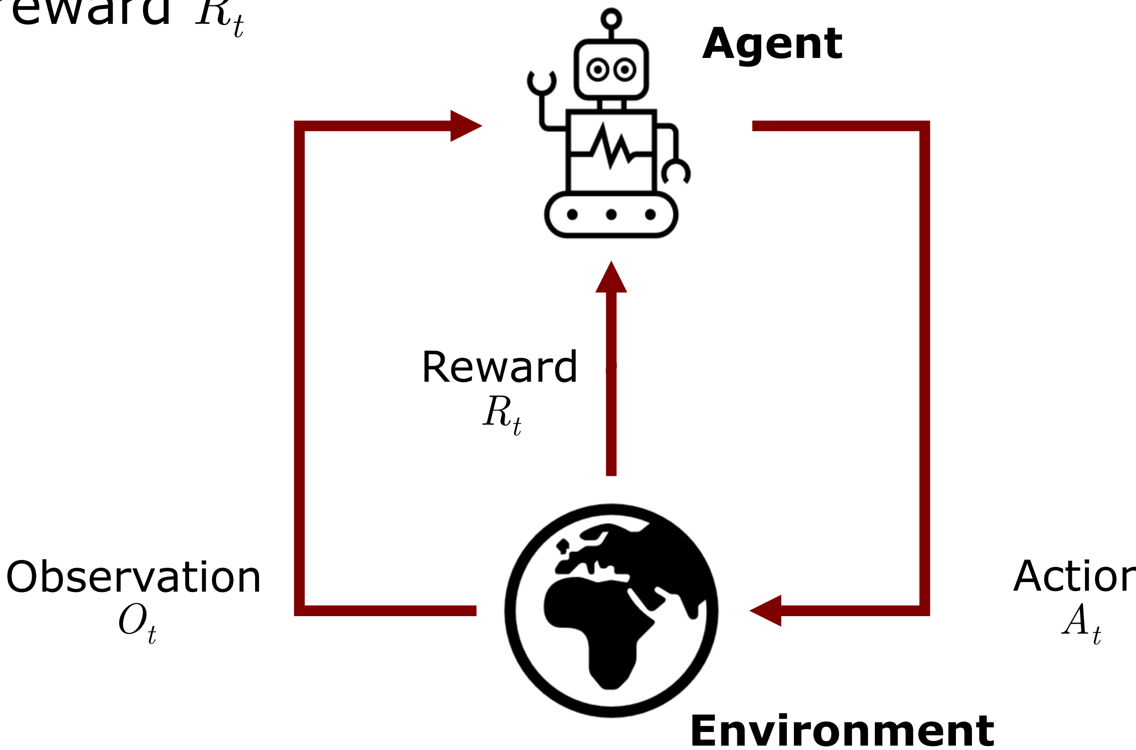
Sequential Decision-Making

- **Goal**: Choose actions to maximise the total expected future reward
- **Challenges**:
  - Subsequent actions depend on what is observed
  - Reward may be delayed



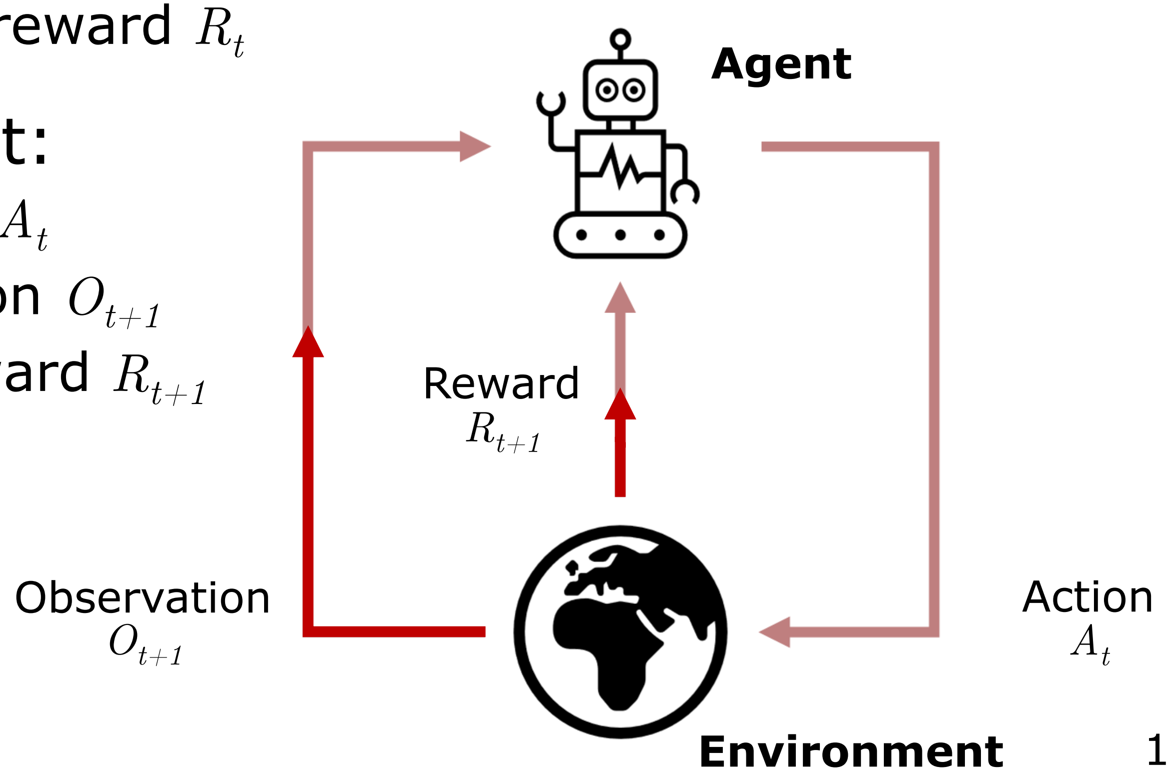
Agent and Environment

- Agent and environment interact continually
- At each time step t , the agent:
  - Performs action A_t
  - Receives observation O_t
  - Receives scalar reward R_t



Agent and Environment

- Agent and environment interact continually
- At each time step t , the agent:
  - Performs action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}



Sources of Uncertainty

1. Uncertainty in actions

2. Uncertainty in observations/states

State Transitions

- **State** S: Determine outcomes and provide all the information for choosing the next decision
- **Transition model** $T(s_{t+1}, a_t, s'_t) = p(s'_{t+1} | s_t, a_t)$: Probability that action a in state s leads to next state s'

State Transitions

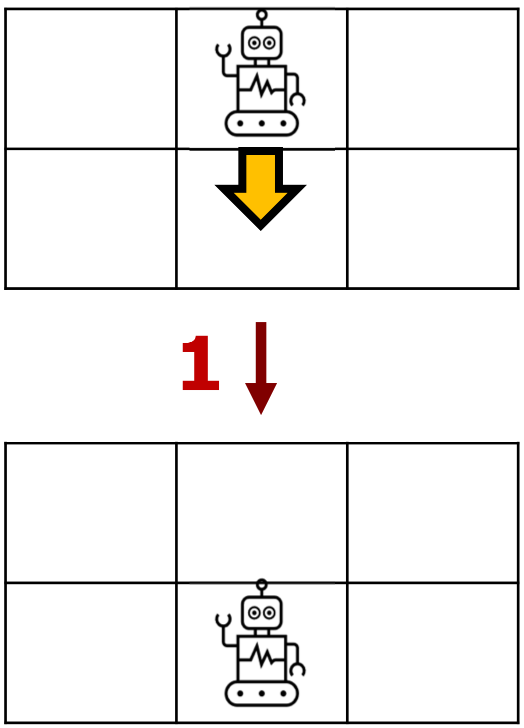
- **State** S: Determine outcomes and provide all the information for choosing the next decision
- **Transition model** $T(s_{t+1}, a_t, s'_t) = p(s'_{t+1} | s_t, a_t)$: Probability that action a in state s leads to next state s'
- **Markov property**: Transition probabilities only depend on the current state and action:

$$p(s_{t+1} \mid a_0, a_1, \ldots, a_t, s_0, s_1, \ldots, s_t) = p(s_{t+1} \mid a_t, s_t)$$

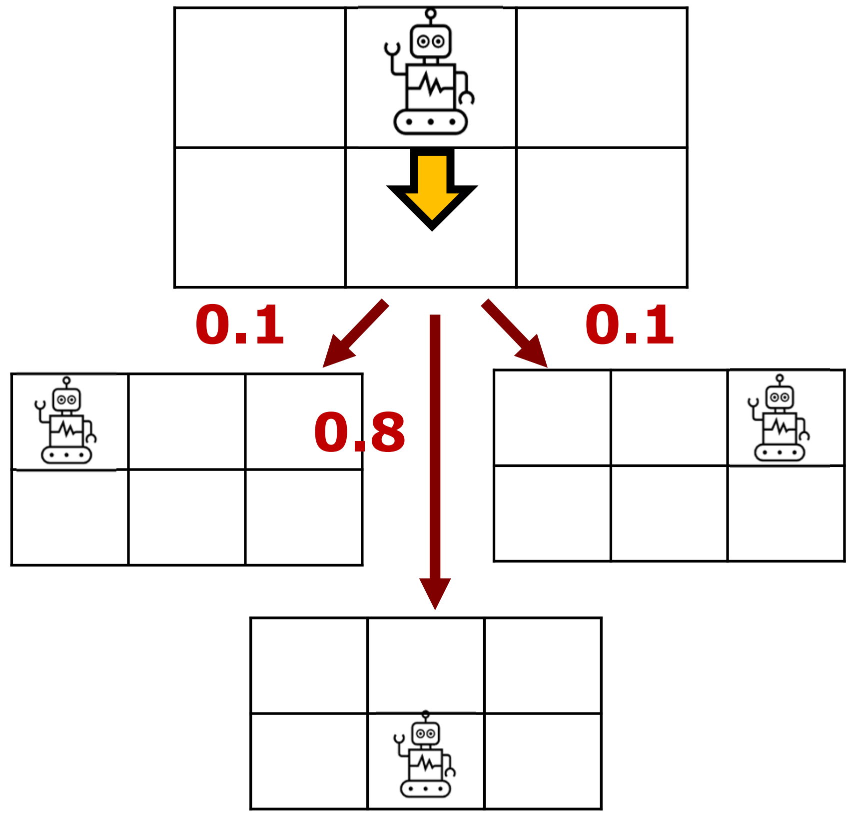
State Transitions

Deterministic

$$p(s'|s,a) = \delta_{s_{\text{next}}}(s') = \begin{cases} 1 & \text{for } s' = s_{\text{next}} \\ 0 & \text{otherwise} \end{cases}$$

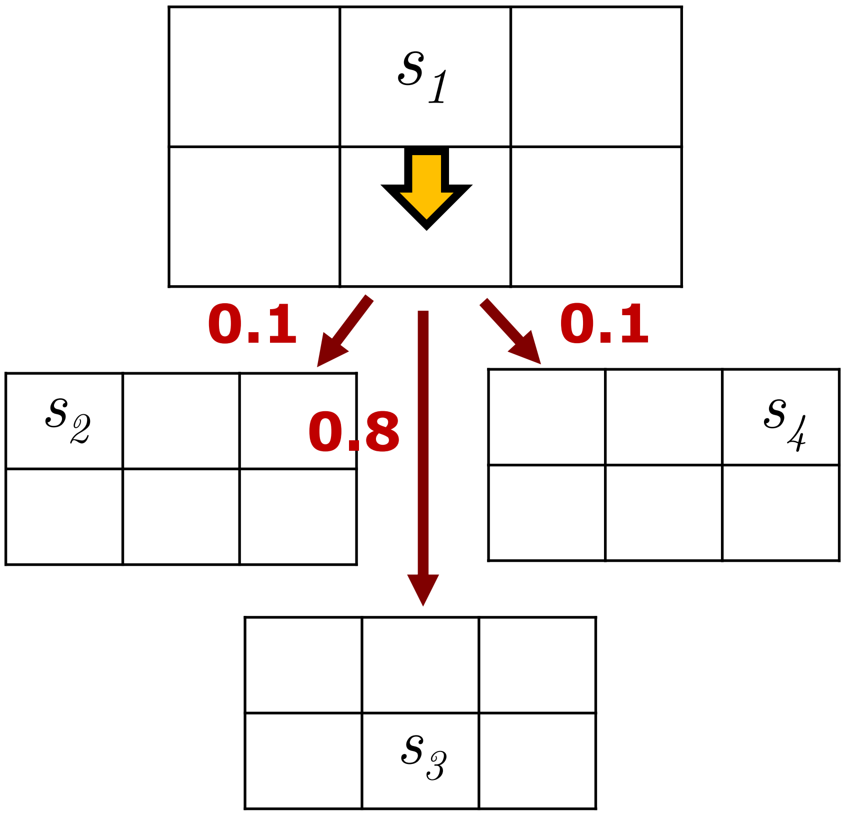


Stochastic



State Transitions

Stochastic



Transition function at s_1:

s'_t	$T(s,a,s')$
s_2	0.1
s_3	0.8
s_4	0.1

Markov Decision Process (MDP)

Markov Decision Process (MDP)

UNIVERSITÄT BONN

Markov Decision Process (MDP)

UNIVERSITÄT BONN

Markov Decision Process (MDP)

Markov Decision Process (MDP)

- General sequential decision-making framework

- Assumptions:
  - Fully observable environment
  - Markov dynamics
  - Stochastic transitions
  - Stationary

MDP Definition

- Defined by a tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$
  - $\mathcal{S}$: finite set of states
  - $\mathcal{A}$: finite set of actions
  - $\mathcal{P}$: state transition probabilities

$$T(s_{t+1},\ a_t,\ s'_t) = p(s'_{t+1} |\ s_t,\ a_t)$$

  - $\mathcal{R}$: reward function

$$\mathbb{E}(R_{t+1} \mid S_t = s, A_t = a)$$

Returns

- **Goal**: Maximise expected cumulative reward (**return** G_t)
  - Episodic vs. continuous tasks

- **Episodic task**
  - Finite horizon, T steps

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T$$

Returns

- **Continuing task**
  - Infinite horizon, never-ending

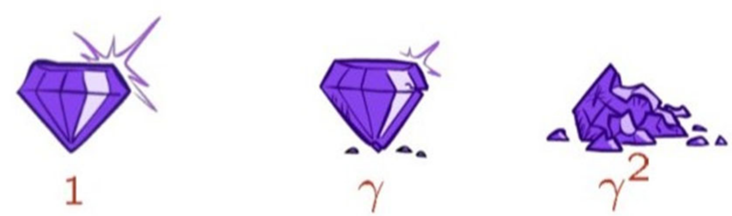
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

  - γ is the **discount factor**

$\gamma = 0$ myopic agent

$\gamma \to 1$ farsighted agent

  - **Why?**
    - Convergence
    - Uncertainty of future reward
    - Humans prefer immediate reward



Policy

- **Policy** π : Determines agent’s behaviour
- Maps states to actions
- Goal of MDP is to find a good policy – maximise returns

- **Deterministic policy** $a = \pi(s)$
- **Stochastic policy** $\pi(a|s) = p(A = a \mid S = s)$

Policies vs. Plans

- **Policies** are more general than **plans**

- **Plan**:
  - Sequence of actions
  - Cannot react to unexpected outcomes
- **Policy**:
  - Tells which action to take from *any* state

Value Functions

- **State-value functions**: expected return starting from state s and following policy π :

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

- **Action-value functions**: expected return starting from state s and following policy:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$
$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

Bellman Equation

- State-value function = immediate reward + discounted state-value of successor state

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}[\underbrace{R_{t+1}}_{\text{immediate reward}} + \underbrace{\gamma v(S_{t+1})}_{\text{discounted value of next state}} \mid S_t = s]
\end{aligned}
$$

Bellman Equation

- Action-value function = immediate reward + discounted action-value of successor state

State-value

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

Action-value

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s]$$

Optimality

Optimal value of a state

$$v_*(s) = \max_\pi v_\pi(s)$$

Optimal value of a state-action pair

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Optimal policy

π_* is an optimal policy if and only if:

$$\pi_*(a|s) > 0 \text{ only where } q_*(s,a) = \max_b q_*(s,b) \qquad \forall s \in \mathcal{S}$$

Solving MDPs –
Dynamic Programming

UNIVERSITÄT BONN

Solving MDPs –
Dynamic Programming

Solving MDPs –
Dynamic Programming

Preliminaries

- **Dynamic programming**: Method for solving sequential problems
- To solve a complex problem:
  - Break down into subproblems
  - Solve the subproblems
  - Combine solutions to subproblems

Requirements

- **Principle of optimality**
  - Optimal substructure
- **Overlapping subproblems**
  - Subproblems recur many times

- MDPs satisfy both requirements
  - Principle of optimality → Bellman equation
  - Overlapping subproblems → value functions

Solving for the Optimal Policy

- Bellman's equation is non-linear

$$v_*(s) = \max_{\pi} v_\pi(s)$$

$$v_*(s) = \max_{\pi}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$v_*(s) = \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathbb{E}_{p(s'|s,a)} \gamma v_*(s')$$

- Solution cannot be found in closed form → need iterative methods
  - Value iteration
  - Policy iteration

Value Iteration

- **Value iteration** applies the optimal value function operator iteratively

$$v_*(s) = \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathbb{E}_{p(s'|s,a)} \gamma v_*(s')$$

  - Start with initial function value guess v_0 and $k=0$ and repeat until convergence

$$v_{k+1} = \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \, \mathbb{E}_{p(s'|s,a)} v_k(s')$$

- It can be shown that value iteration converges

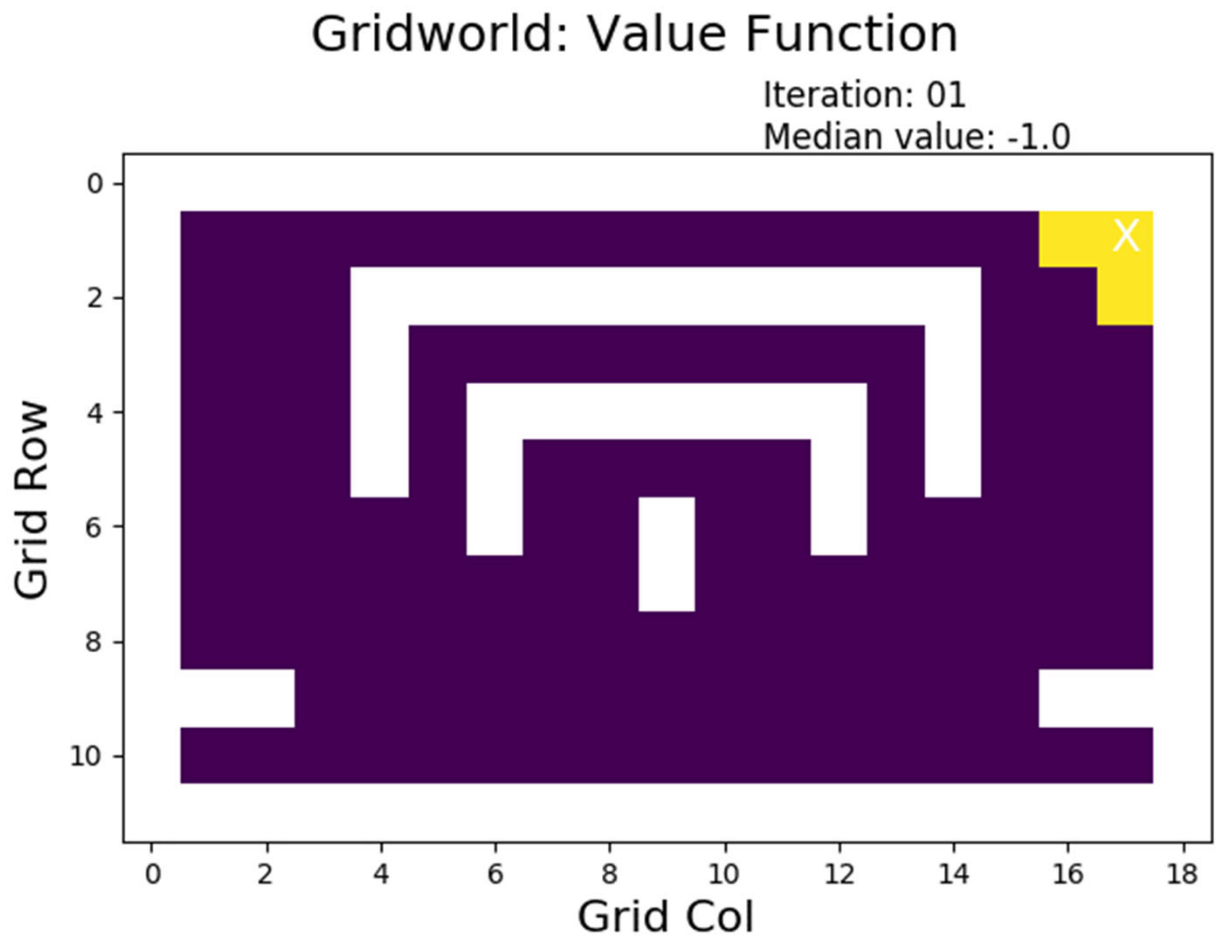
Policy Iteration

- The agent only cares about finding the optimal policy (not all the state-values)
- **Policy iteration** alternates the following steps, starting with an initial policy π_0 :
  - **Policy evaluation**: given policy π_k , calculate $v_k(s) = v_{\pi_k}$, $s \in \mathcal{S}$
  - **Policy improvement**: calculate maximum expected utility policy π_{k+1} :

$$\pi_{k+1} = \arg\max_{a \in \mathcal{A}} q_\pi(s, a)$$

Example

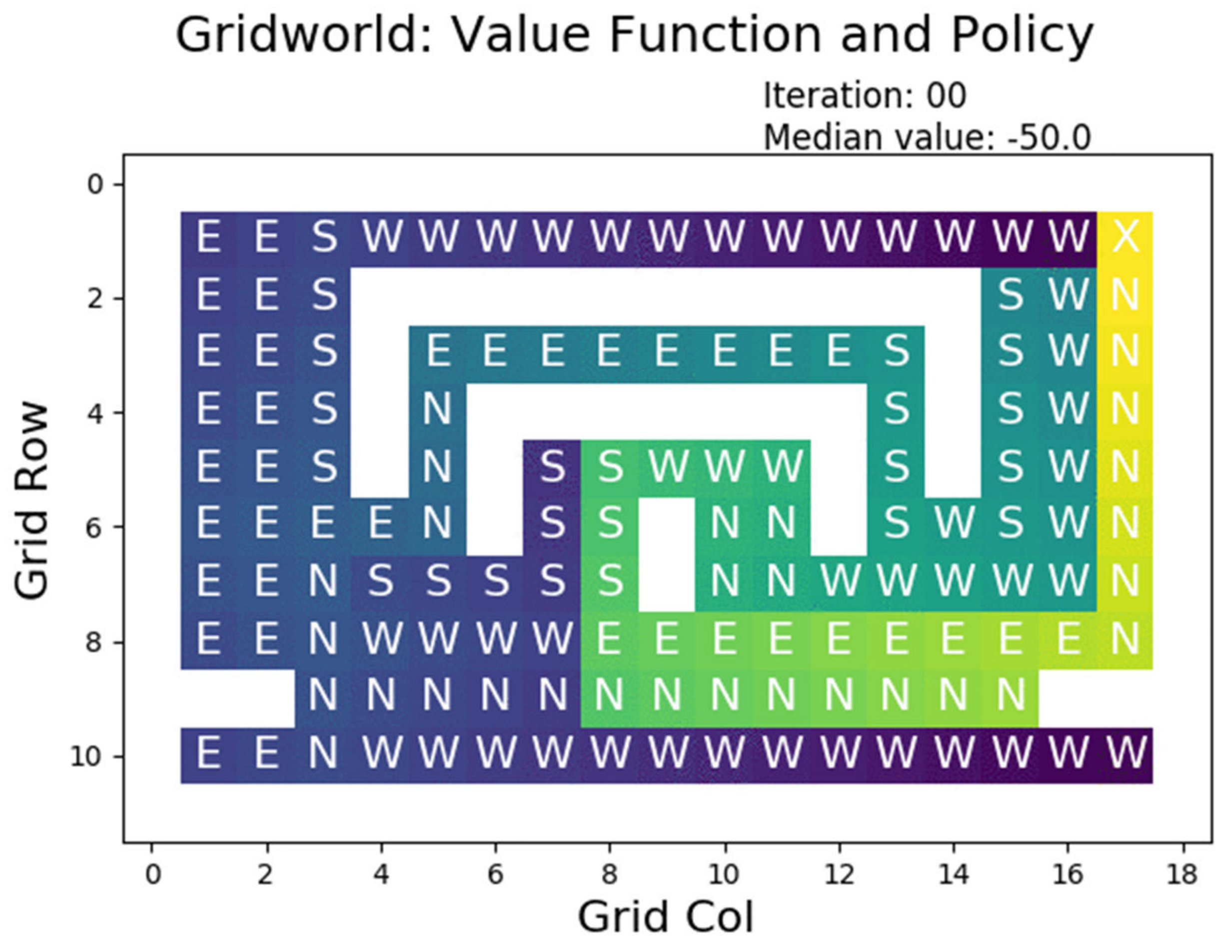
- Grid world: value iteration



https://www.datascienceblog.net/post/reinforcement-learning/mdps_dynamic_programming/

Example

- Grid world: policy iteration



Summary

- Decision-making problem
  - Aim: Maximise expected cumulative reward
- Markov Decision Processes (MDPs)
  - Bellman's equation
- Dynamic Programming
  - Value iteration
  - Policy iteration

Further Reading

- Markov Decision Process (fabioconcina.github.io)
- Navigating in Gridworld using Policy and Value Iteration - Data Science Blog: Understand. Implement. Succed.
- Reinforcement Learning : Markov-Decision Process (Part 1) | by blackburn | Towards Data Science
- Introduction to Reinforcement Learning with David Silver | DeepMind
- Dynamic Programming and Optimal Control – Dimitri Bertsekas (1976)
  - Textbook: Dynamic Programming and Optimal Control (athenasc.com)

Thank you for your attention