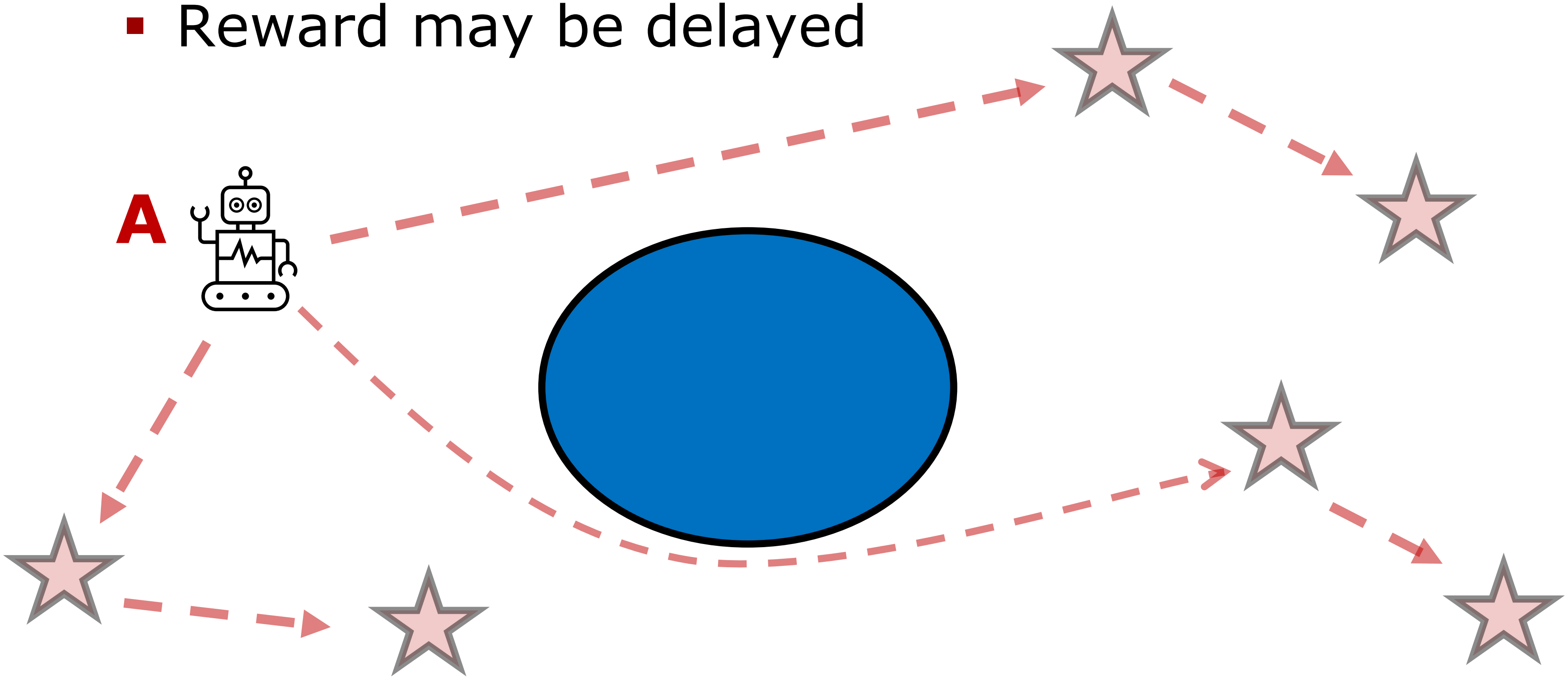
7 – Decision-Making Under Uncertainty

Dr. Marija Popović

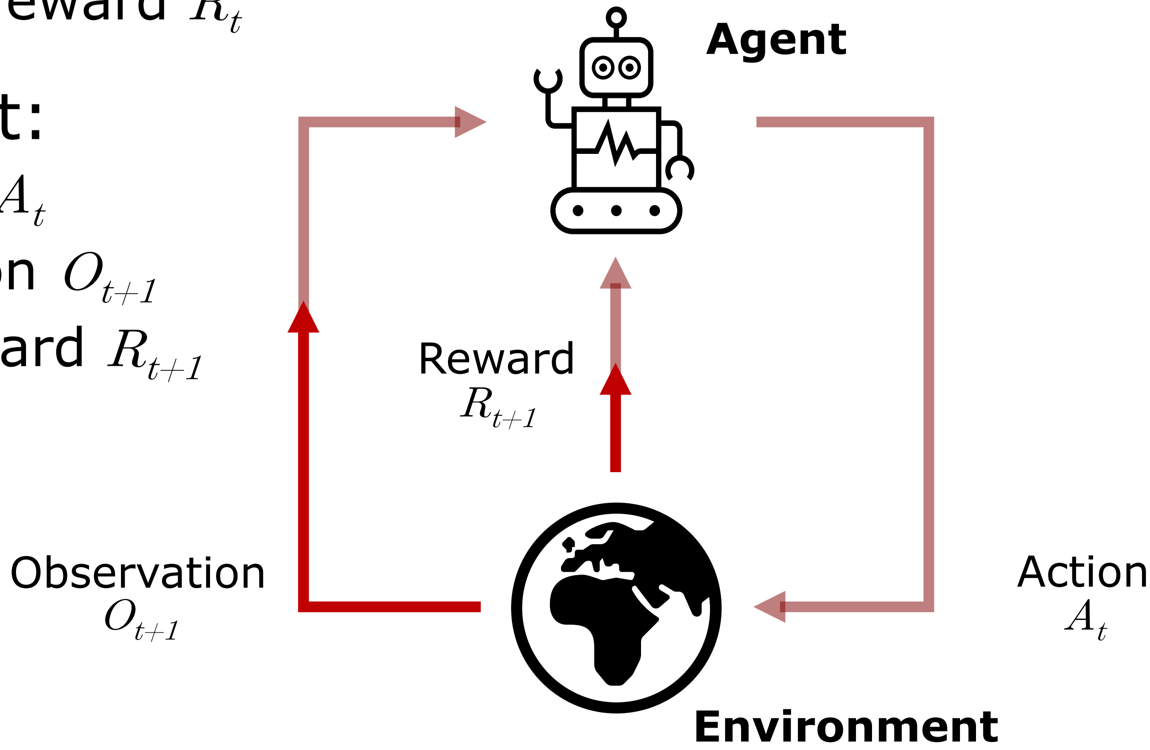
Sequential Decision-Making

- **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
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}



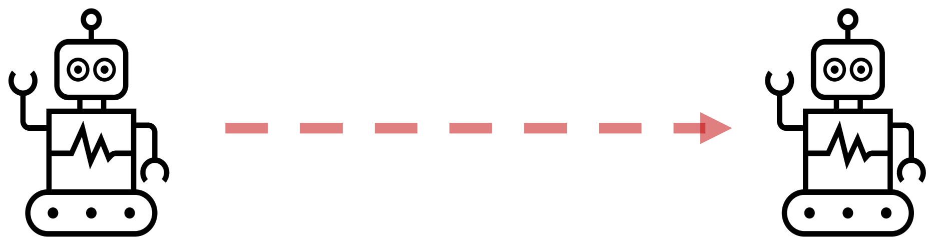
Sources of Uncertainty

- Changes in the environment over time
- Actions take time to execute
- Actions can fail
- Actuators are noisy
- Sensors are limited in range/resolution and noisy/imprecise
- Imperfect knowledge by the agent
- Modelling errors
- …

Probabilistic decision-making

Sources of Uncertainty

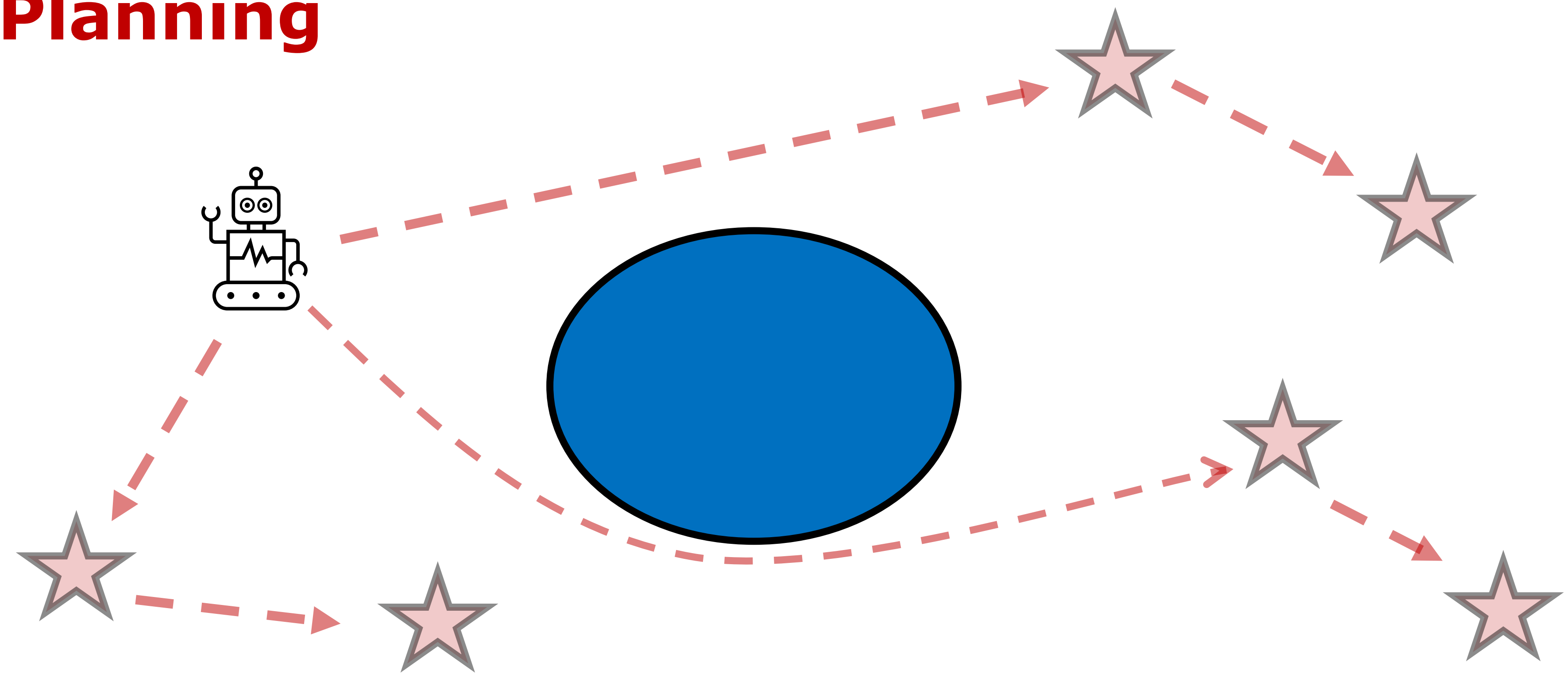
1. Uncertainty in actions



2. Uncertainty in observations/states

Sources of Uncertainty

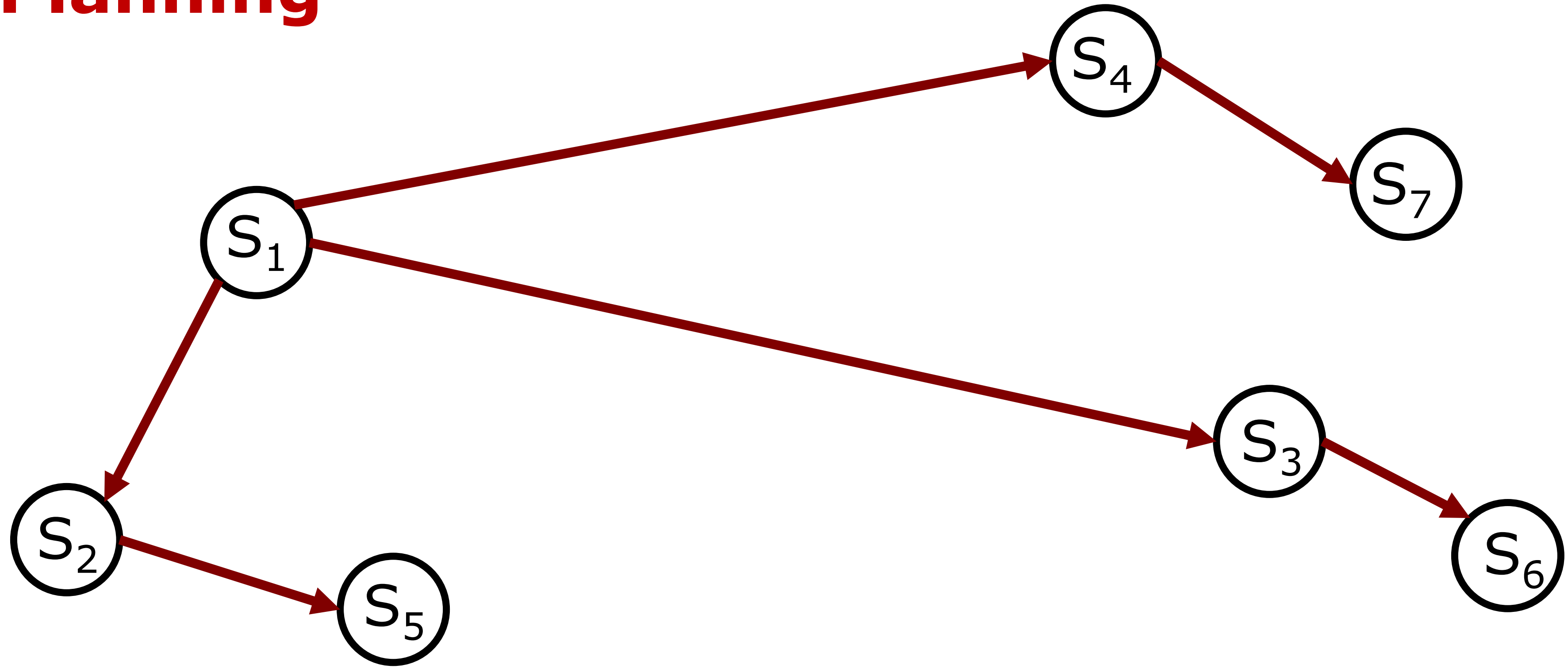
No uncertainty → **Planning**



Use a graph $G = (V, E)$

Sources of Uncertainty

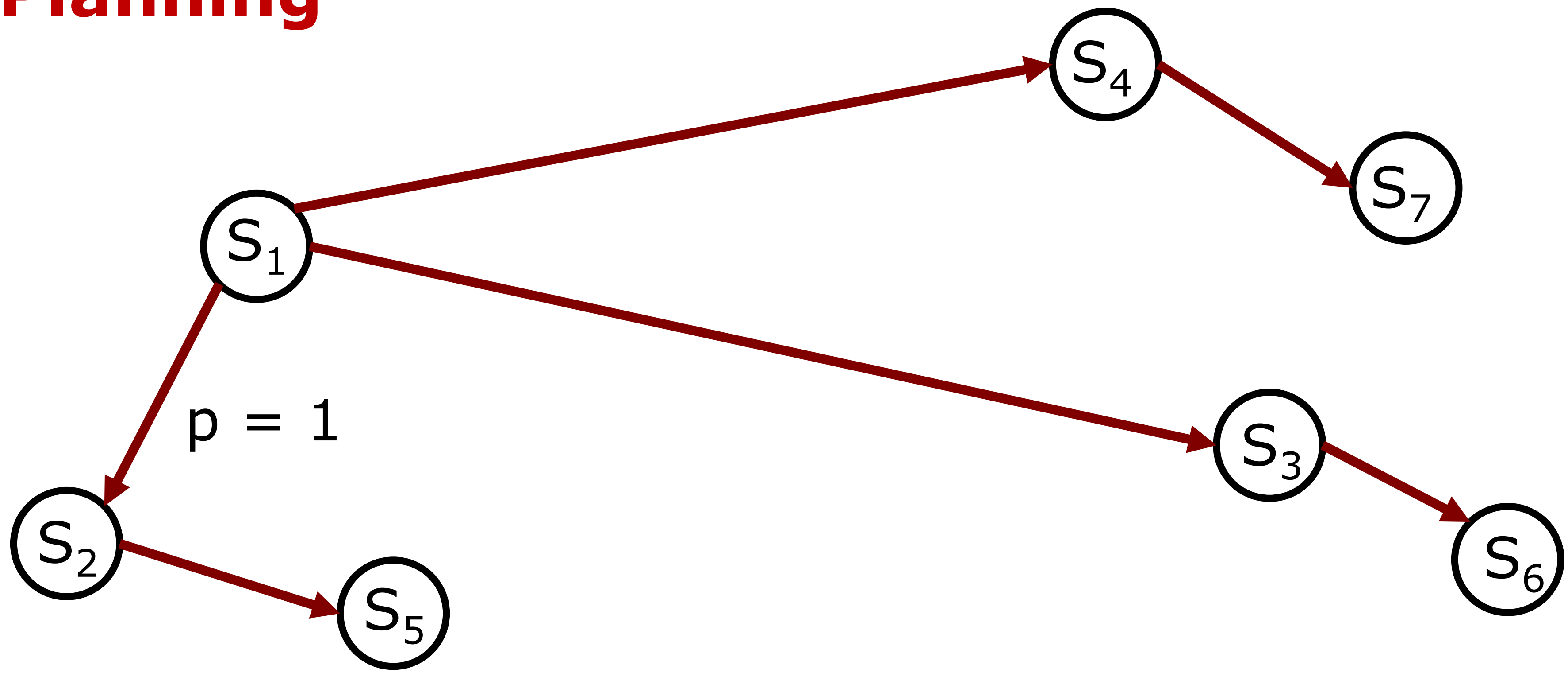
No uncertainty → **Planning**



Use a graph $G = (V, E)$

Sources of Uncertainty

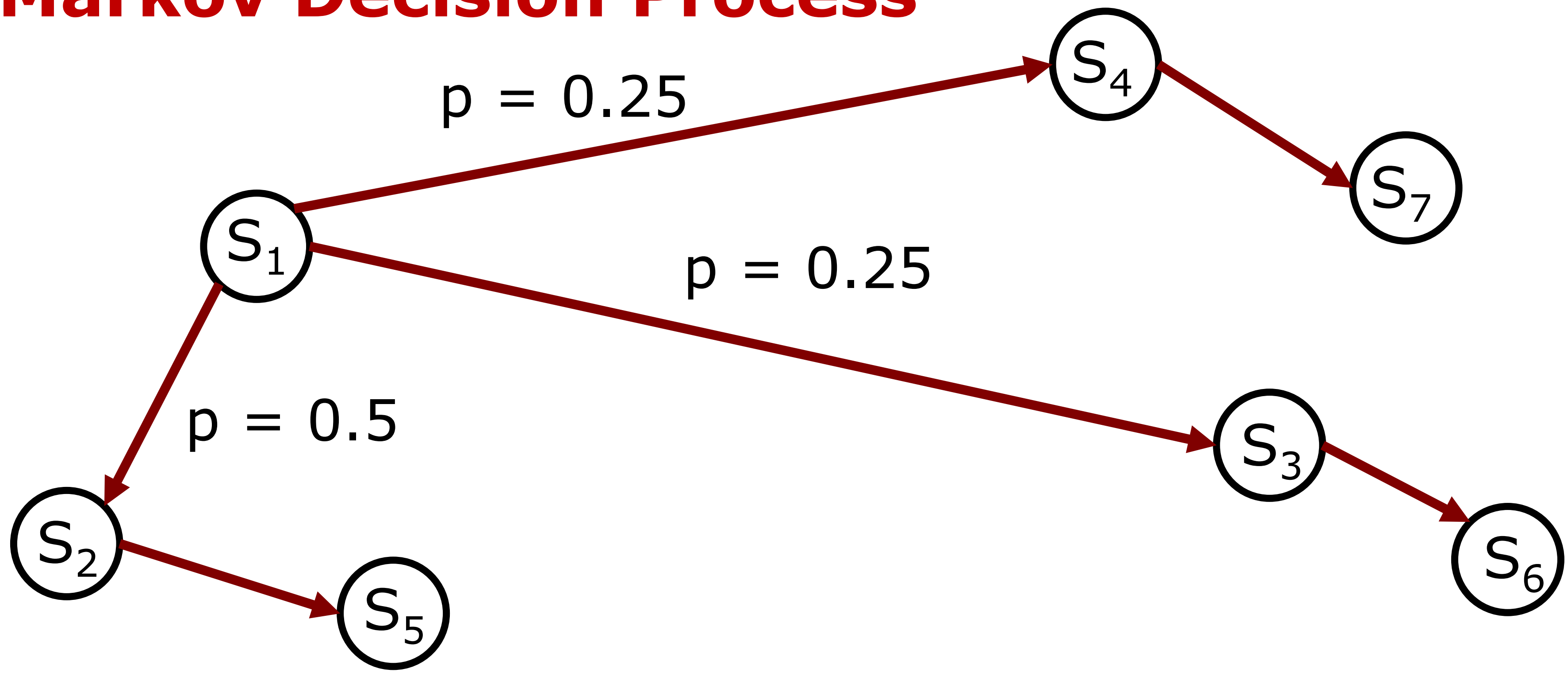
No uncertainty → **Planning**



Use a graph $G = (V, E)$

Sources of Uncertainty

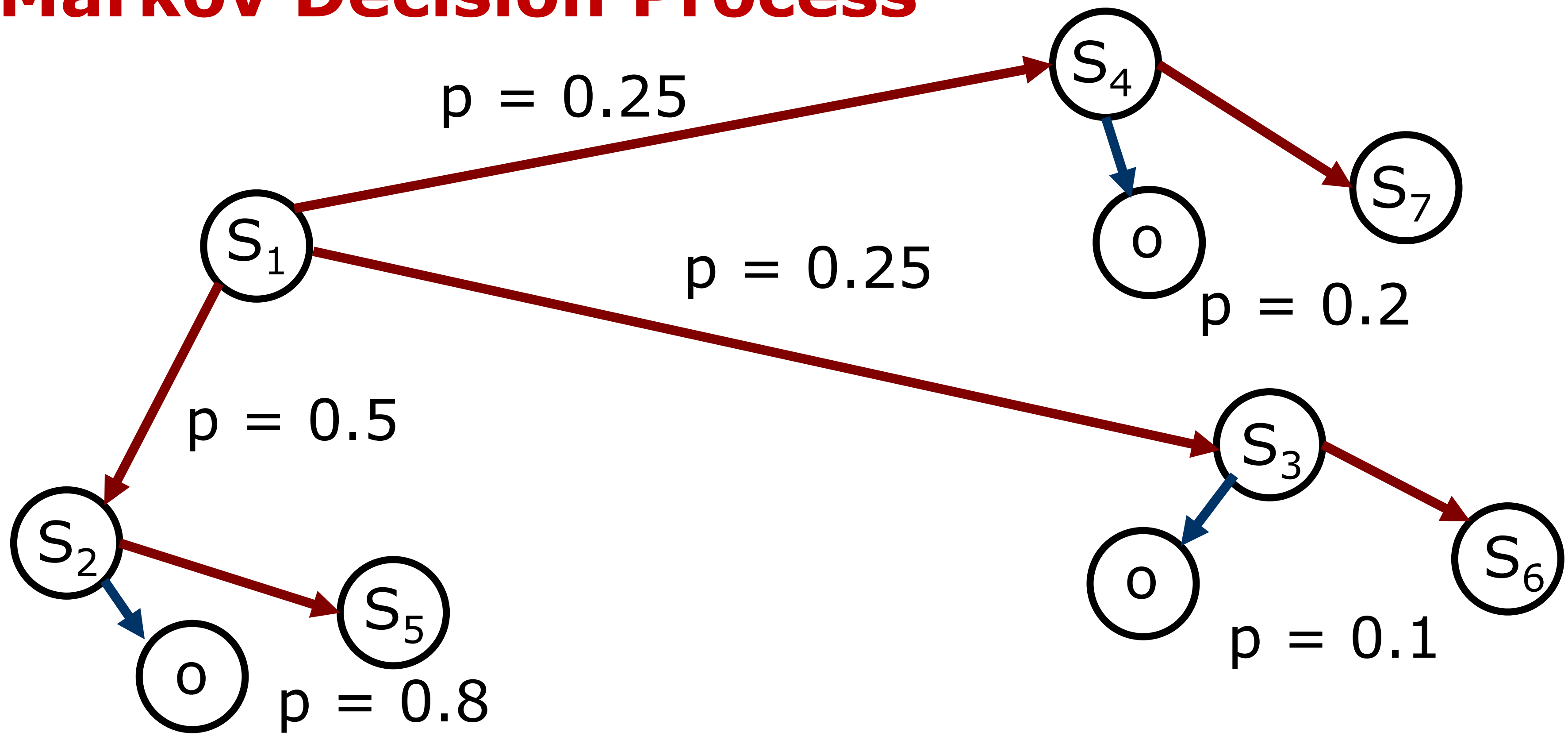
Actuation uncertainty →
Markov Decision Process



Use MDP framework $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$

Sources of Uncertainty

State uncertainty → **Partially Observable Markov Decision Process**



Use POMDP framework $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}>$

Partially Observable Markov Decision Process (POMDP)

Markov Decision Process (MDP)

- General sequential decision-making framework

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

MDP Definition

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

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

  - $\mathcal{R}$: reward function

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

POMDP Definition

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

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

  - $\mathcal{R}$: reward function

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

  - Ω: finite set of observations
  - $\mathcal{O}$: observation probabilities

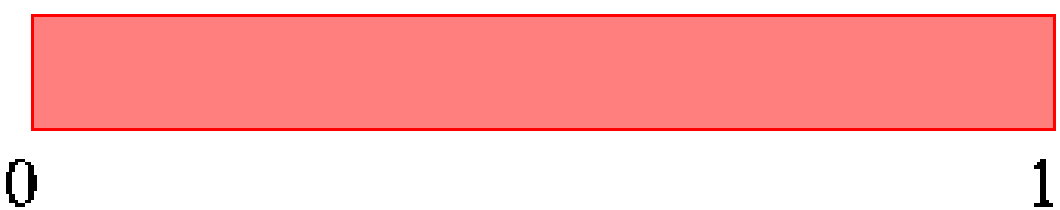
$$O(s'_{t+1}, a_t, o_{t+1}) = p(o_{t+1} \mid s'_{t+1}, a_t)$$

Belief State Space

- **Belief state**: Probability distribution over states

$$b(s) = p(s)$$

- **Belief state space**: Set of all possible probability distributions



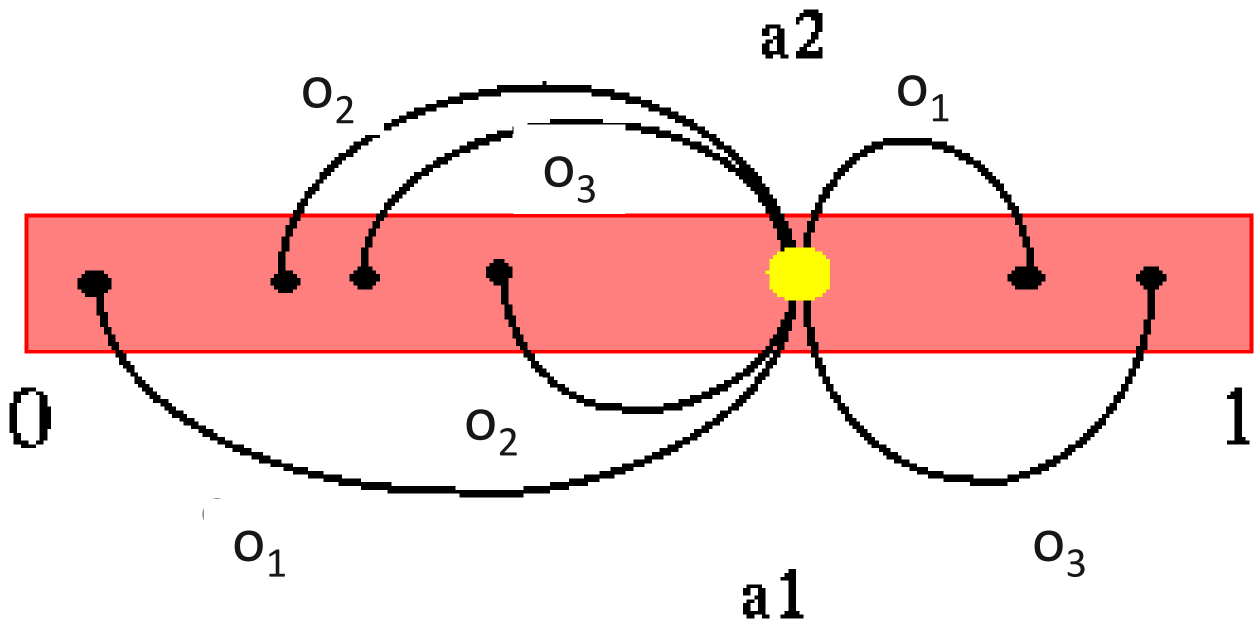
$$b(s_1) = p$$

$$b(s_2) = 1 - p$$

http://cs.brown.edu/research/ai/pomdp/tutorial/pomdp-solving.html

Belief Update

- How to recursively update the belief?



http://cs.brown.edu/research/ai/pomdp/tutorial/pomdp-solving.html

More clearly stated: if we take an action and get an observation, then we know with certainty what

Belief Update

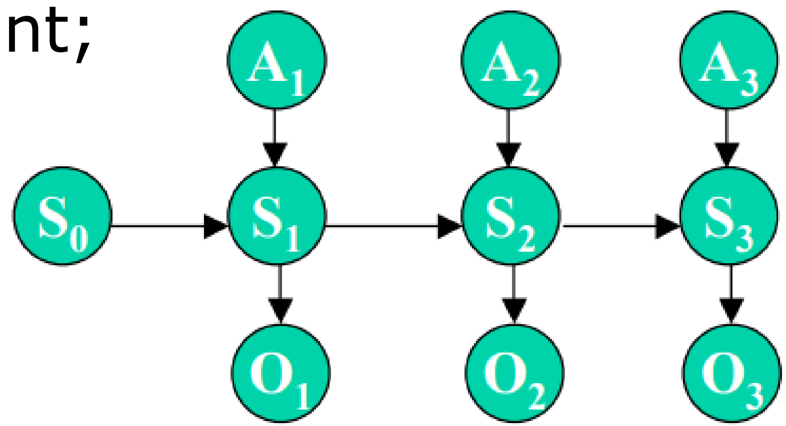
- How to recursively update the belief?

- Use a **filtering** procedure:

$$b'(s') = \eta O(o \,|\, s', a) \sum_{s \in \mathcal{S}} T(s' \,|\, s, a) b(s)$$

where η is a normalising constant; inverse of:

$$\sum_{s' \in \mathcal{S}} O(o \,|\, s', a) \sum_{s \in \mathcal{S}} T(s' \,|s, a) b(s)$$



Belief MDP Definition

- Defined by a tuple: $<\mathcal{B}, \mathcal{A}, \tau, \rho>$
  - $\mathcal{B}$: **infinite** set of **beliefs**
  - $\mathcal{A}$: finite set of actions
  - τ : belief state transition probabilities

$$\tau(b, a, b') = \sum_{o \in \Omega} \underline{p(b' \mid a, b, o)}\, \underline{p(o \mid a, b)}$$

$=1$ if belief update leads to b' ; else $=0$

$$\sum_{s' \in \mathcal{S}} O(o \mid s', a) \sum_{s \in \mathcal{S}} T(s' \mid s, a) b(s)$$

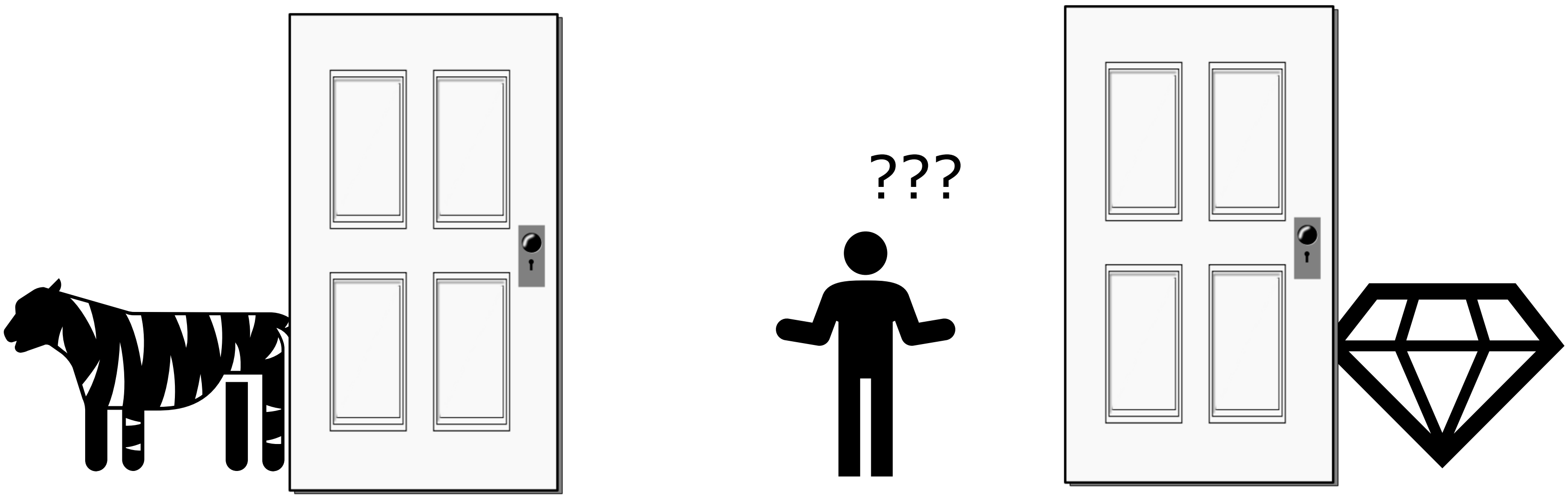
  - ρ : reward function on belief states

$$\rho(s, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$$

Example: Tiger Problem

Example: Tiger Problem

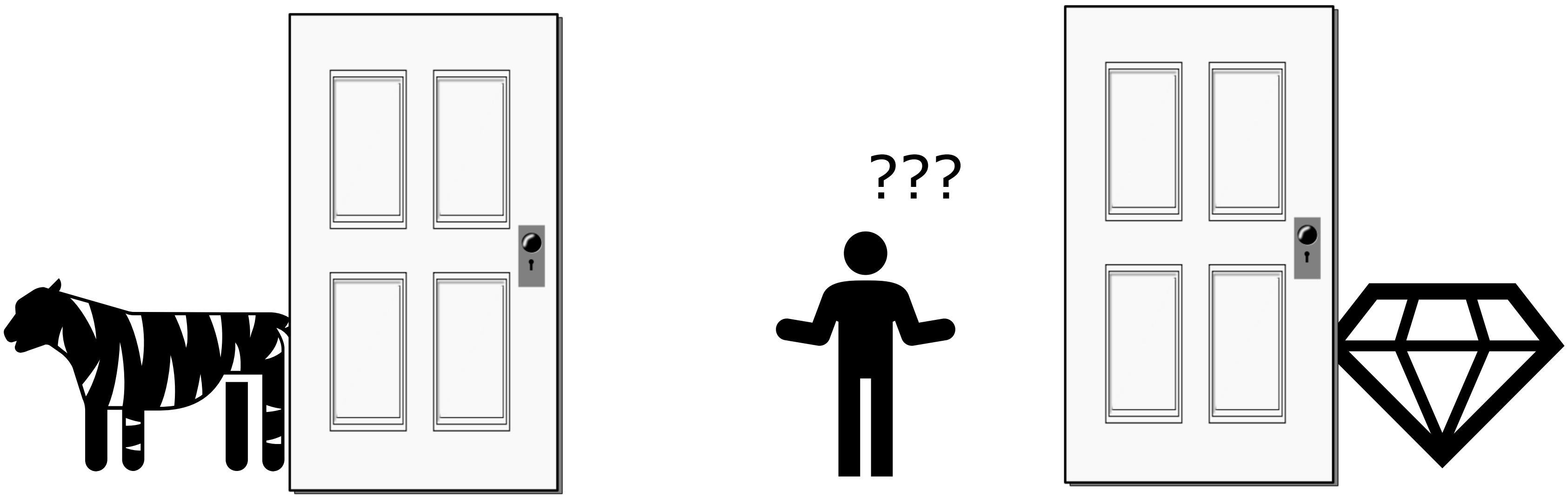
- Standing in front of two closed doors
- Open door with tiger: **- reward**; open door with treasure: **+ reward**
- Can also **listen** for tiger



Kaelbling et al. (1998), “Planning and acting in partially observable stochastic domains,“ in: Artificial intelligence. 101(1-2): pp. 99-134.

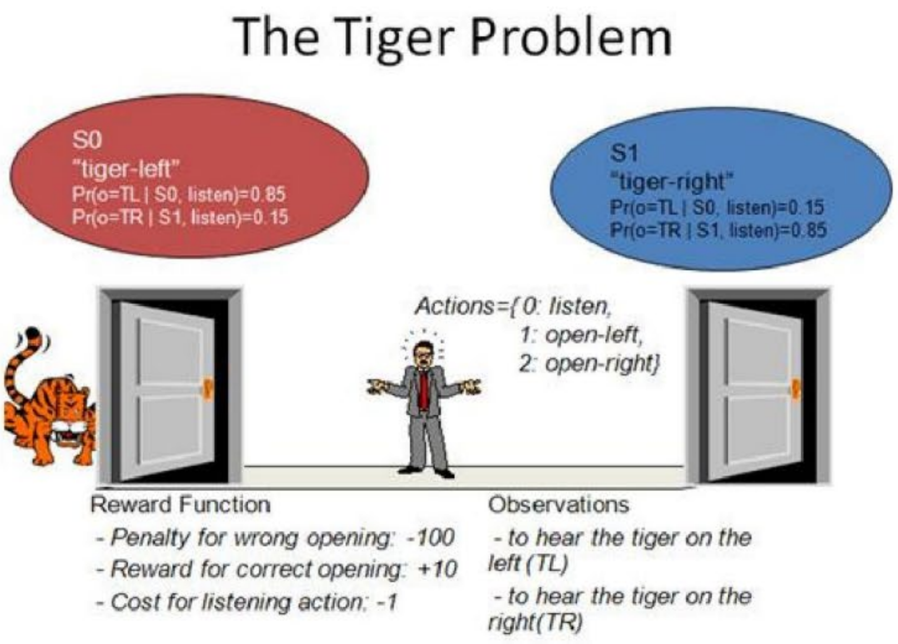
Example: Tiger Problem

- **States**: {tiger-left, tiger-right}
- **Actions**: {listen, open-left, open-right}
  - Transitions: no change (listen), restart (open)
- **Observations**: {hear-tiger-left, hear-tiger-right}
- **Rewards**: tiger, treasure, listening



Kaelbling et al. (1998), “Planning and acting in partially observable stochastic domains,“ in: Artificial intelligence. 101(1-2): pp. 99-134.

Example: Tiger Problem



Listening does not change the position of the tiger

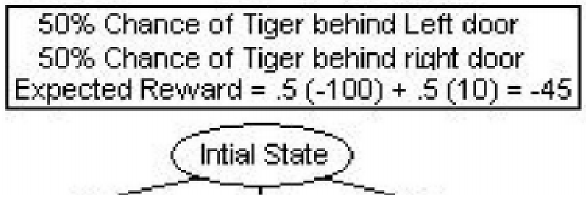
Prob. (LISTEN)	Tiger: left	Tiger: right
Tiger: left	1.0	0.0
Tiger: right	0.0	1.0

Position of the tiger resets after we open a door

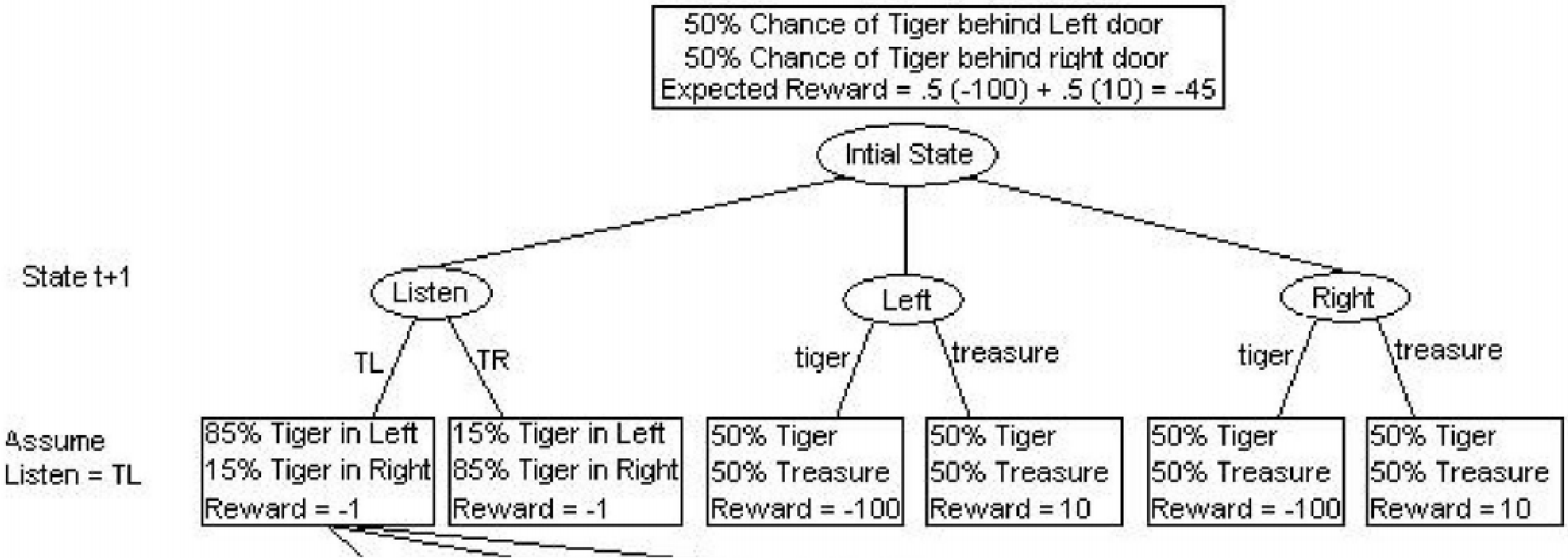
Prob. (LEFT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Prob. (RIGHT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

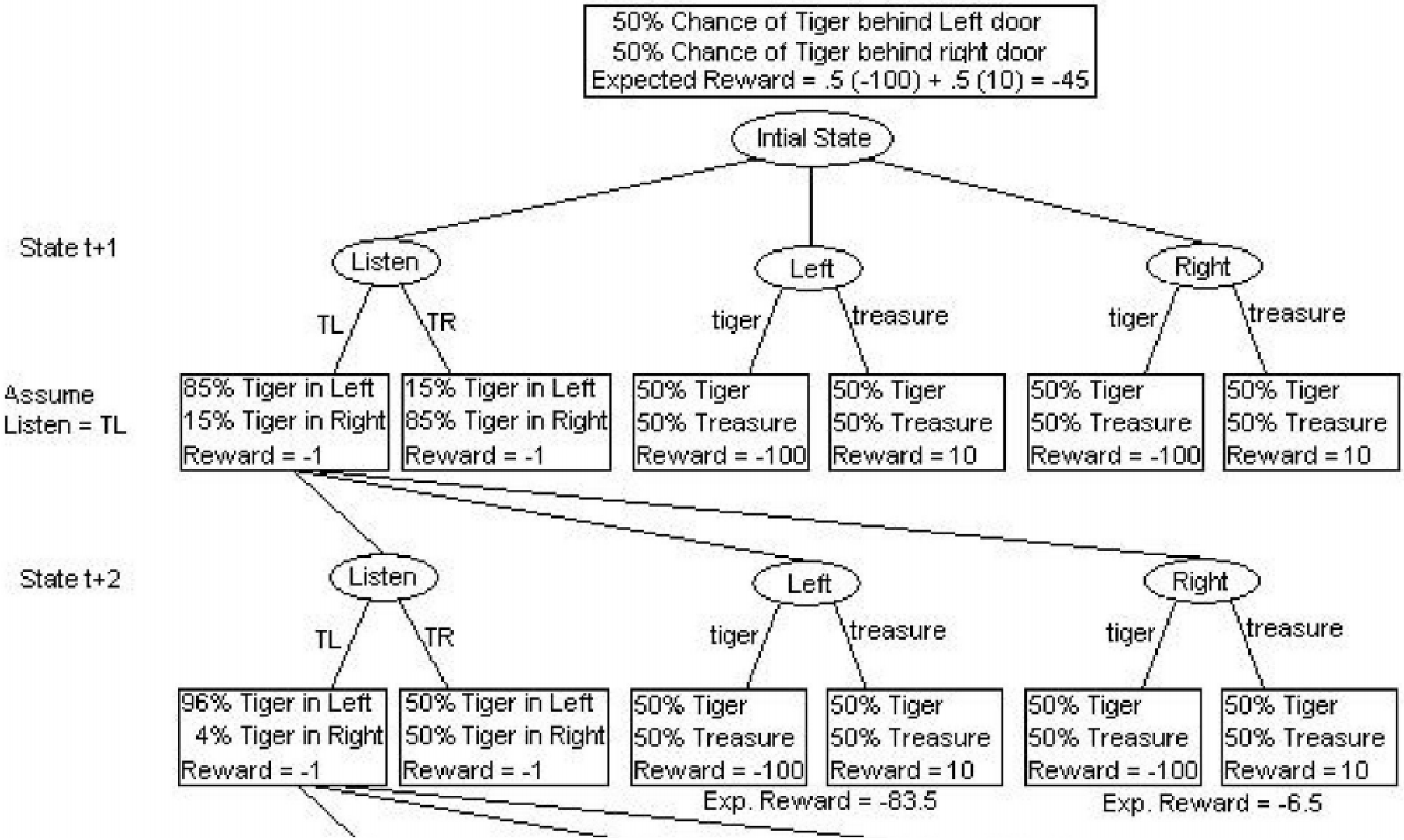
https://robotics.cs.rutgers.edu/wp-content/uploads/2020/04/pomdps.pdf



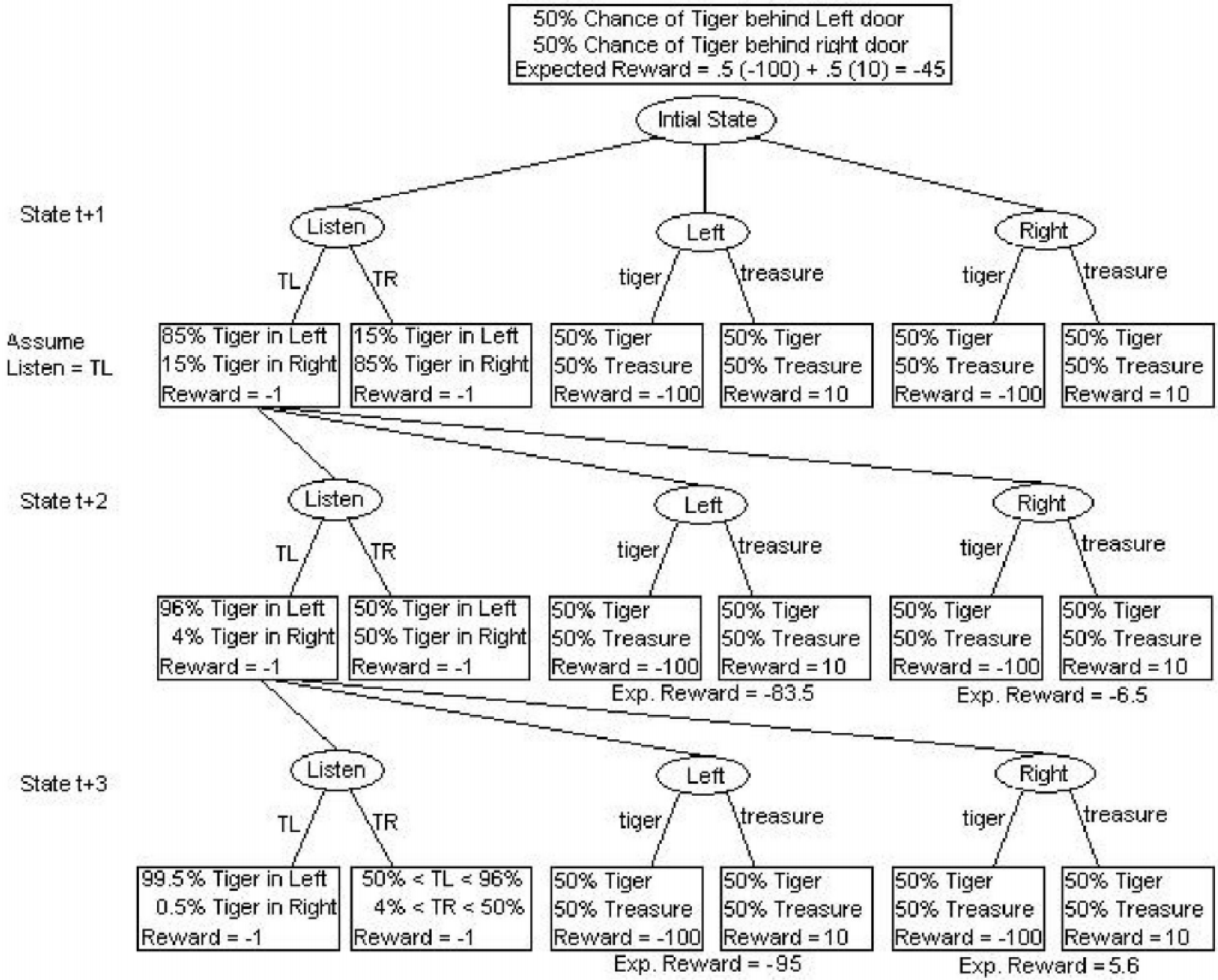
https://robotics.cs.rutgers.edu/wp-content/uploads/2020/04/pomdps.pdf



https://robotics.cs.rutgers.edu/wp-content/uploads/2020/04/pomdps.pdf



https://robotics.cs.rutgers.edu/wp-content/uploads/2020/04/pomdps.pdf



https://robotics.cs.rutgers.edu/wp-content/uploads/2020/04/pomdps.pdf

Solving POMDPs

Belief MDP Policies

- **POMDP policy**: Maps **beliefs** to **actions**

$$\pi(b) = a$$

- **Optimal policy** π^* yields highest expected reward from any belief state

- **Bellman equation** for POMDPs**:**

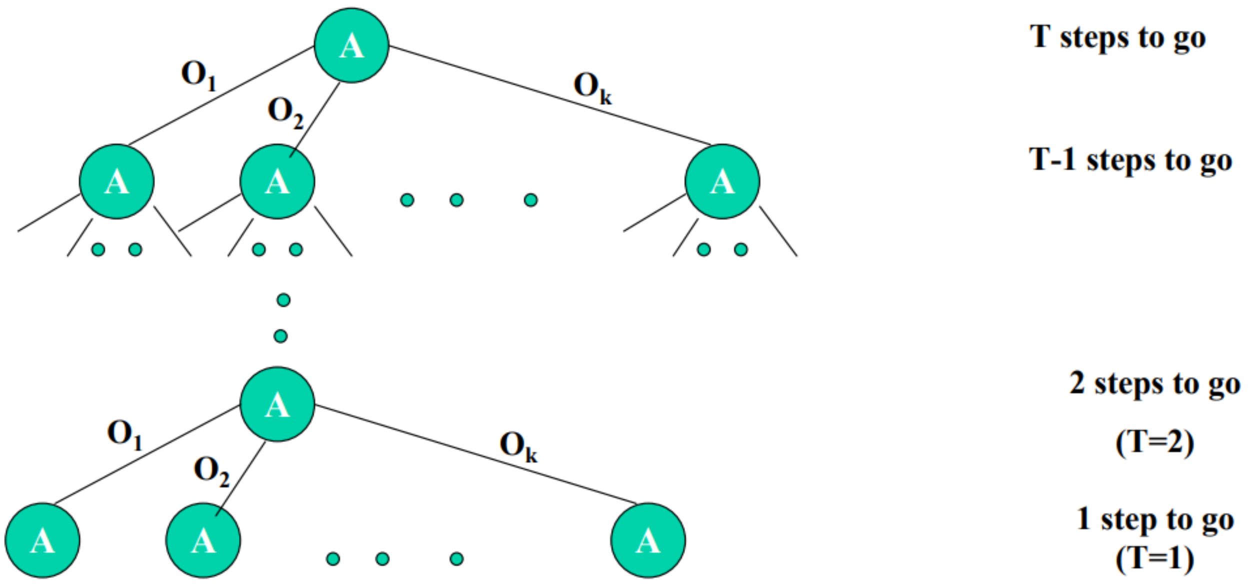
$$V^*(b) = \max_{a \in \mathcal{A}} \left[r(b,a) + \gamma \sum_{o \in \Omega} p(o \mid b, a) V^*(\tau(b,a,o)) \right]$$

Continuous-space → **Very hard to solve!**

"Forward Search" Method

- **General approach**:
  - Search over sequences with limited look-ahead
  - Branching over actions and observations

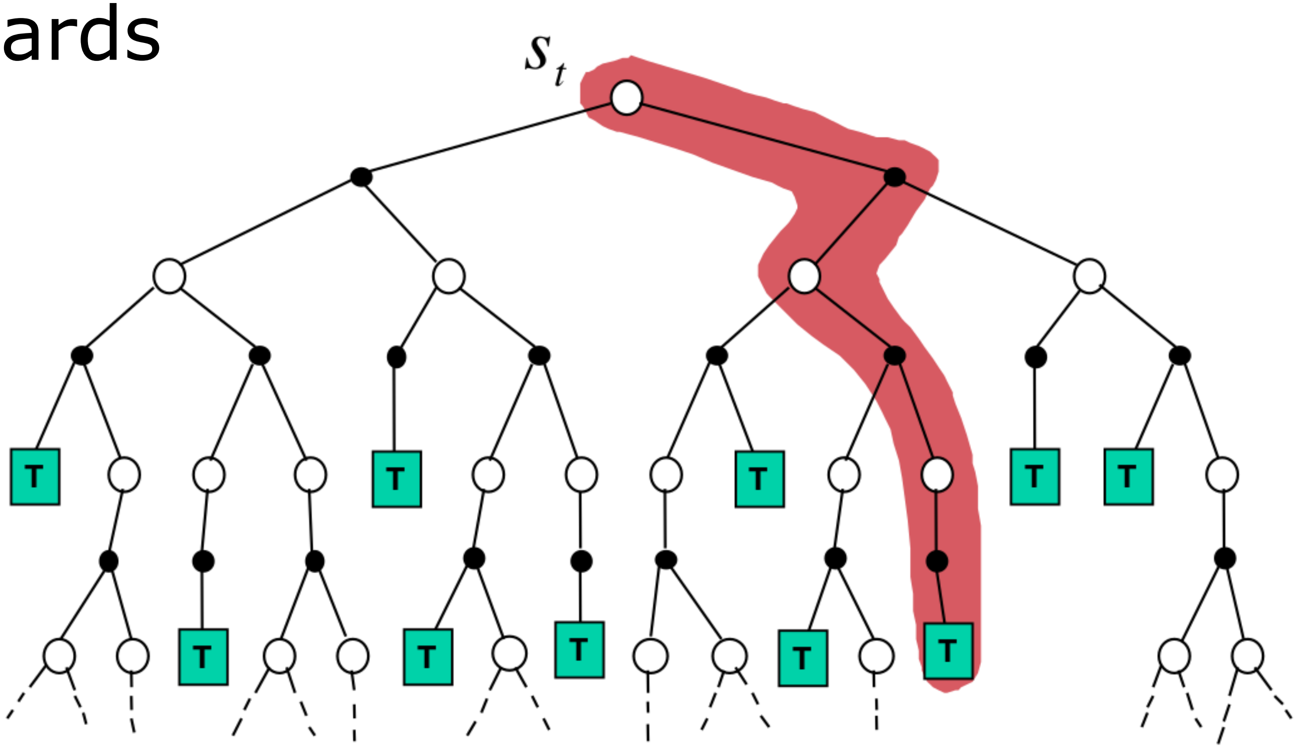
t-step
policy tree



https://www.techfak.uni-bielefeld.de/~skopp/Lehre/STdKI_SS10/POMDP_tutorial.pdf

Monte Carlo Tree Search (MCTS)

- **Completely observable** MDP
- Build search tree of state-action sequences
- **Rollout**: simulate episodes and collect rewards



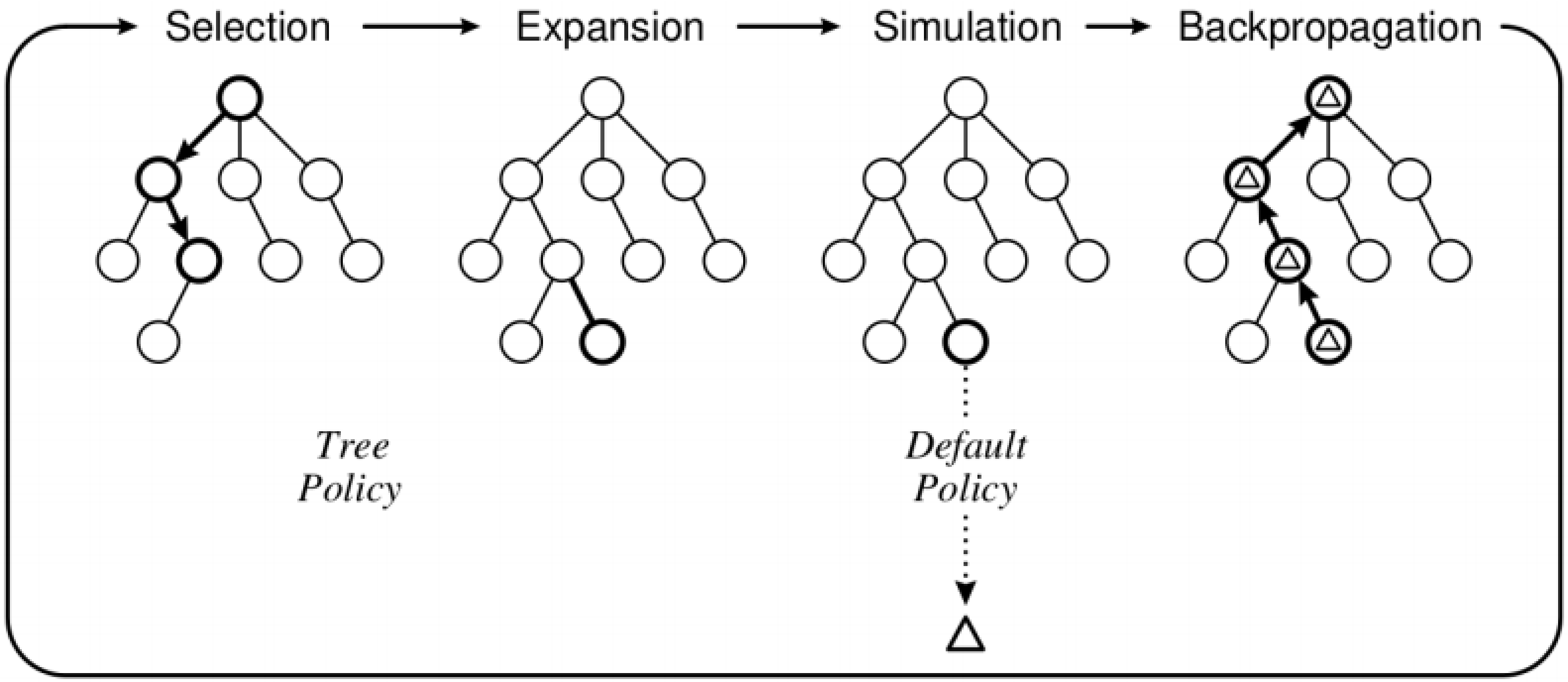
Upper Confidence Tree Search

- Which action to select to maximise information about the problem space?
- Trade-off between exploration and exploitation while growing the tree

- **Upper confidence bound**

$$Q(s, a, i) = \frac{1}{N(s, a, i)} \sum_{k=1}^{K} \sum_{u=t}^{T} \mathbb{1}(i \in epi.k) G_k(s, a, i) + c\sqrt{\frac{ln(n(s))}{n(s, a)}}$$

MCTS Algorithm

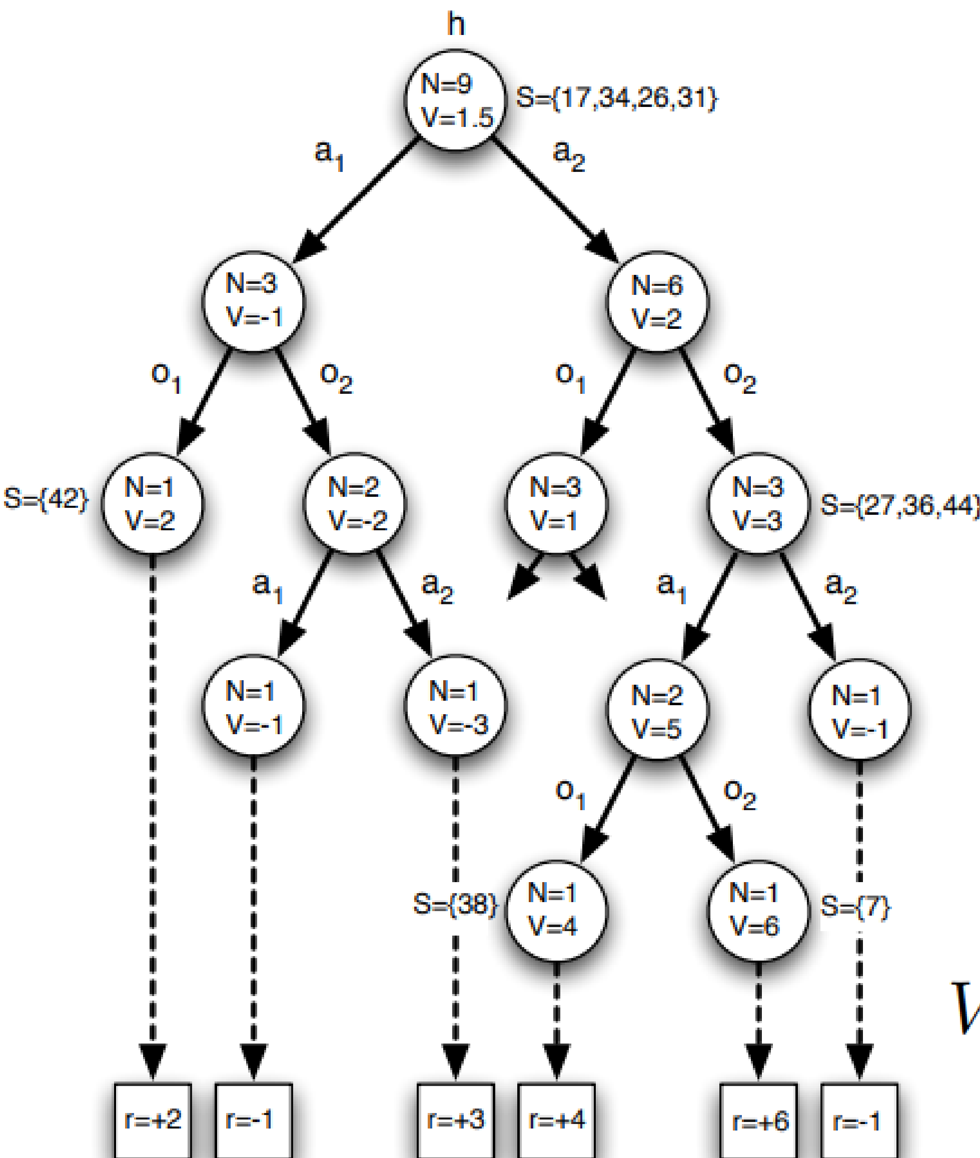


Partially Observable Monte Carlo Policy (POMCP)

- How to handle partial observability?

- Use a simulator to receive observations from chosen actions
- Store histories of observation-action sequences in each node instead of the states

POMCP Tree



- POMCP structure analogous to MCTS
- Use rollout simulator
- Actions are selected trading off exploration and exploitation

$$V^{\oplus}(ha) = V(ha) + c\sqrt{\frac{\log N(h)}{N(ha)}}$$

Silver, D. and Veness, J. (2010). "Monte-Carlo planning in large POMDPs", In: NIPS.

Other POMDP solution methods

- Exact algorithms (very inefficient)
- Heuristics based on underlying MDP
- Point-based algorithms
- Recurrent neural networks
- …

Summary

- Decision-making under uncertainty
- Partially Observable Markov Decision Processes (POMDPs)
  - Belief state space
  - Belief MDPs
- Solution methods
  - Monte Carlo Tree Search
  - Partially Observable Monte Carlo Policy

Further Reading

- POMDPs for Dummies: Page 1 (brown.edu)
- pomdp_py Documentation — pomdp_py 1.0 documentation (h2r.github.io)
- POMDP: Introduction to Partially Observable Markov Decision Processes (r-project.org)
- https://www.techfak.uni-bielefeld.de/~skopp/Lehre/STdKI_SS10/POMDP_tutorial.pdf
- pomdps.pdf (rutgers.edu)
- POMDPs: Who Needs them?
- Silver, D. and Veness, J. (2010). “Monte-Carlo planning in large POMDPs”, In: NIPS.
- Dynamic Programming and Optimal Control – Dimitri Bertsekas (1976)
  - Textbook: Dynamic Programming and Optimal Control (athenasc.com)

Thank you for your attention

Belief Update Derivation

A belief state b is a probability distribution over $\mathcal{S}$. We let $b(s)$ denote the probability assigned to world state s by belief state b. The axioms of probability require that $0 \le b(s) \le 1$ for all $s \in \mathcal{S}$ and that $\sum_{s \in \mathcal{S}} b(s) = 1$. The state estimator must compute a new belief state, b', given an old belief state b, an action a, and an observation o. The new degree of belief in some state s', $b'(s')$, can be obtained from basic probability theory as follows:

$$
\begin{aligned}
b'(s') &= \Pr(s'|o,a,b) \\
&= \frac{\Pr(o|s',a,b)\Pr(s'|a,b)}{\Pr(o|a,b)} \\
&= \frac{\Pr(o|s',a)\sum_{s\in\mathcal{S}}\Pr(s'|a,b,s)\Pr(s|a,b)}{\Pr(o|a,b)} \\
&= \frac{O(s',a,o)\sum_{s\in\mathcal{S}}T(s,a,s')b(s)}{\Pr(o|a,b)}
\end{aligned}
$$

Bayes Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

The denominator, $\Pr(o|a,b)$, can be treated as a normalizing factor, independent of s', that causes b' to sum to 1. The state estimation function $SE(b,a,o)$ has as its output the new belief state b'.

POMCP Algorithm

Algorithm 1 Partially Observable Monte-Carlo Planning

procedure SEARCH(h)
  repeat
    if $h = empty$ **then**
      $s \sim \mathcal{I}$
    else
      $s \sim B(h)$
    end if
    SIMULATE($s, h, 0$)
  until TIMEOUT()
  return $\operatorname{argmax}_b V(hb)$
end procedure

procedure ROLLOUT($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return $r + \gamma$.ROLLOUT($s', hao, depth+1$)
end procedure

procedure SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  if $h \notin T$ **then**
    for all $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    end for
    return ROLLOUT($s, h, depth$)
  end if
  $a \leftarrow \operatorname{argmax}_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  $R \leftarrow r + \gamma$.SIMULATE($s', hao, depth + 1$)
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R
end procedure

Silver, D. and Veness, J. (2010). “Monte-Carlo planning in large POMDPs”, In: NIPS.