# Artificial Intelligence

## Planning 5: Markov Decision Processes

Jendrik Seipp

Linköping University

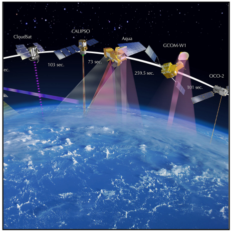
based on slides by Thomas Keller and Malte Helmert (University of Basel)

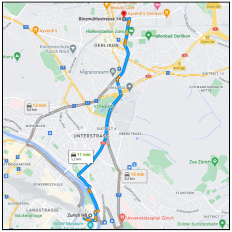
# Intended Learning Outcomes

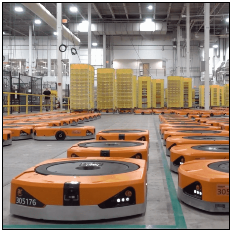
- explain the difference between deterministic and probabilistic planning tasks
- contrast SSPs and discounted reward infinite-horizon MDPs
- explain and use the methods for solving MDPs: LP, PI, VI

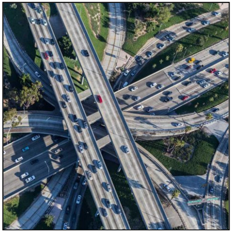
# Motivation

# MDP Examples




# Decisions Under Uncertainty

# (Maximum) Expected Utility


Adam wants to invest in the stock market. He considers the options Bellman Inc. (B), Howard Ltd. (H) and Markov Tec. (M).

A (simplified) expert analysis makes the following predictions:

| Bellman Inc. | Howard Corp. | Markov Tec. |
|---|---|---|
| +2 with 30% | +3 with 40% | +4 with 20% |
| +1 with 60% | ±0 with 10% | +2 with 30% |
| ±0 with 10% | −1 with 50% | −1 with 50% |

What are the expected payoffs (or expected utility / reward)?

# (Maximum) Expected Utility

Adam wants to invest in the stock market. He considers the options Bellman Inc. (B), Howard Ltd. (H) and Markov Tec. (M).

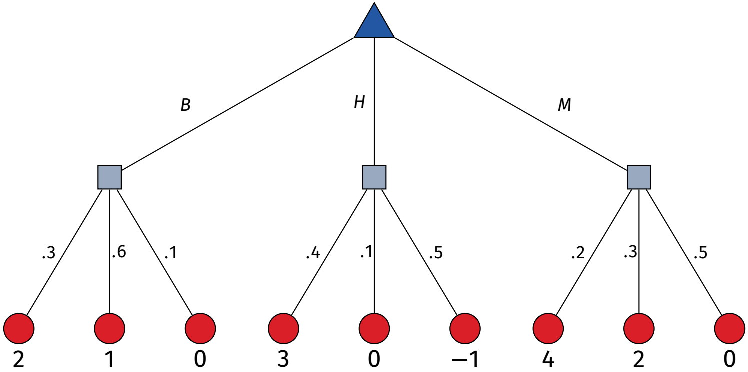
A (simplified) expert analysis makes the following predictions:

| Bellman Inc. | Howard Corp. | Markov Tec. |
|---|---|---|
| +2 with 30% | +3 with 40% | +4 with 20% |
| +1 with 60% | ±0 with 10% | +2 with 30% |
| ±0 with 10% | −1 with 50% | −1 with 50% |

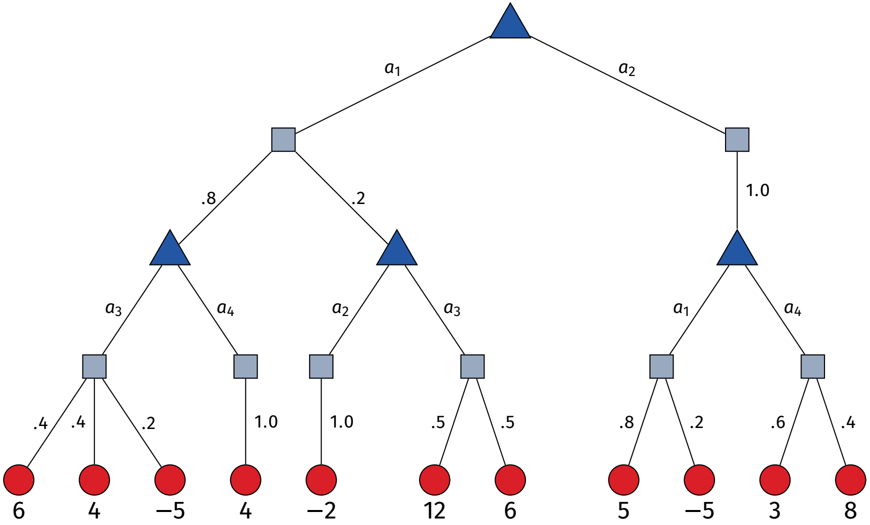
What are the expected payoffs (or expected utility / reward)?

What is the rational decision (if Adam trusts the analysis)?

# Tree Interpretation



# Sequential Example



# Similarity?

Does this remind you of something?

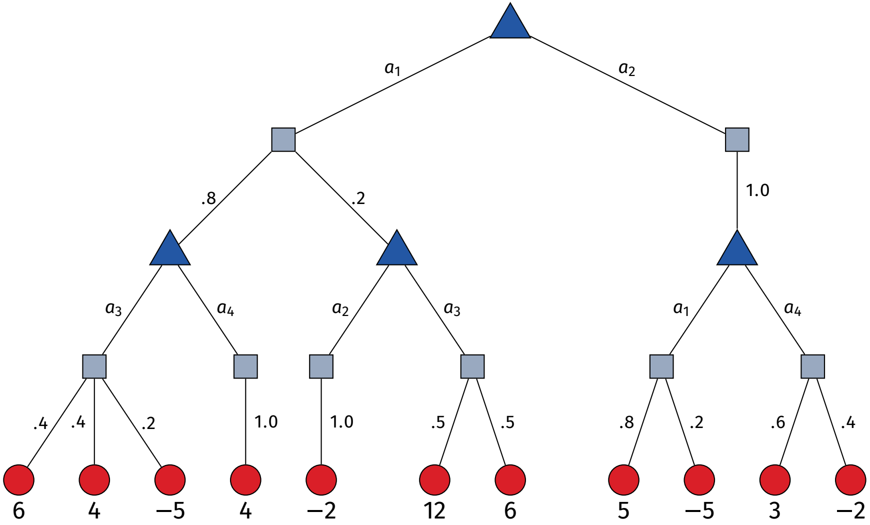
# Expectimax

Does this remind you of something? $\rightsquigarrow$ Minimax

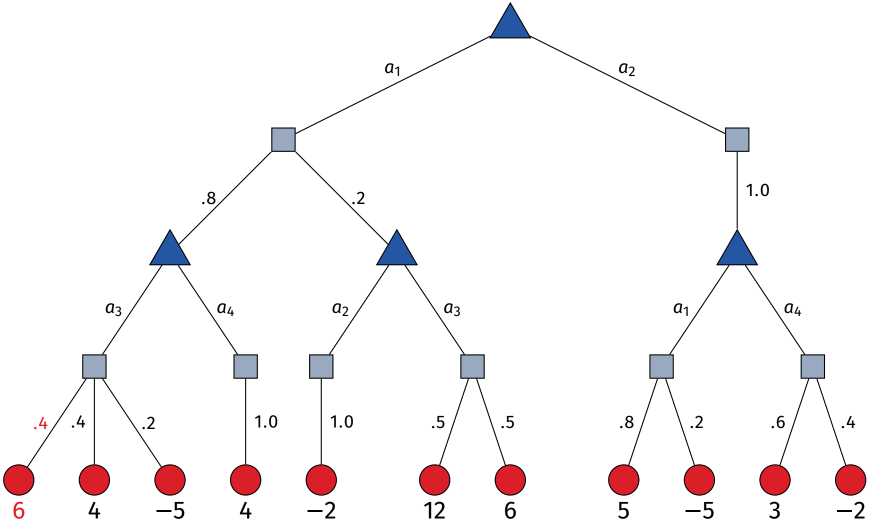
Expectimax is analogous algorithm for
sequential decision making under uncertainty

1. depth-first search through tree
2. apply utility function in terminal state
3. compute utility value of inner nodes
from below to above through the tree:
   - max node: utility is maximum of utility values of children
   - chance node: utility is probability-weighted sum of utility values of children
4. move selection in root:
choose a move that maximizes the computed utility value

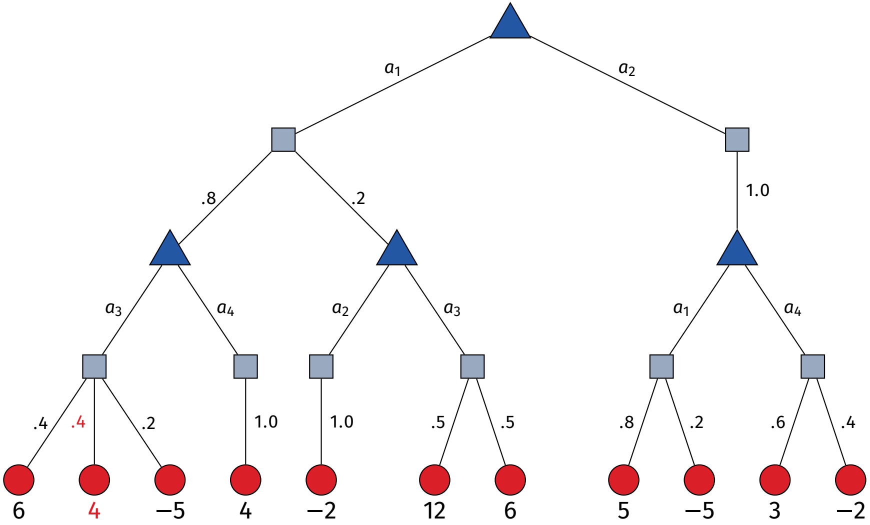
# Expectimax: Example



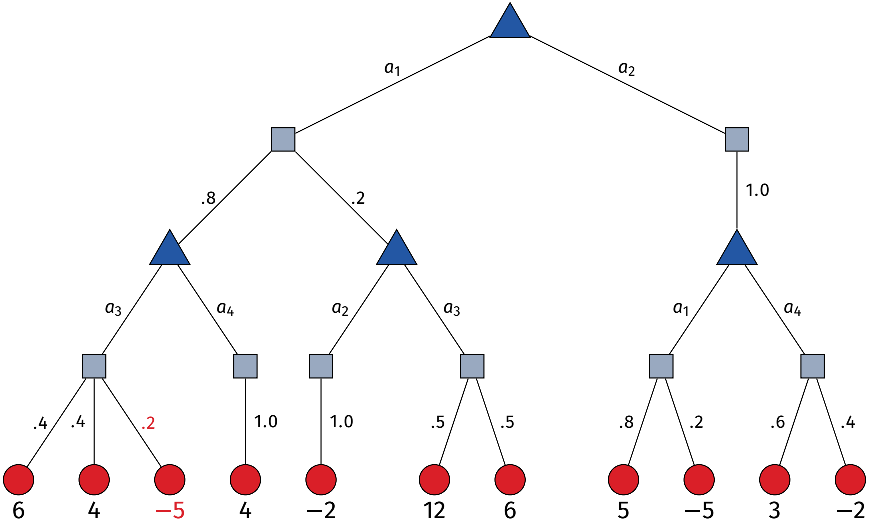
# Expectimax: Example



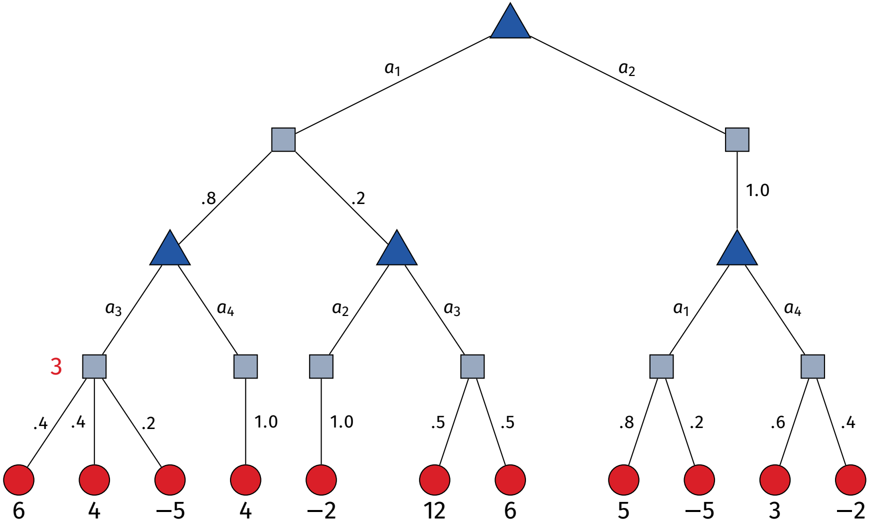
# Expectimax: Example



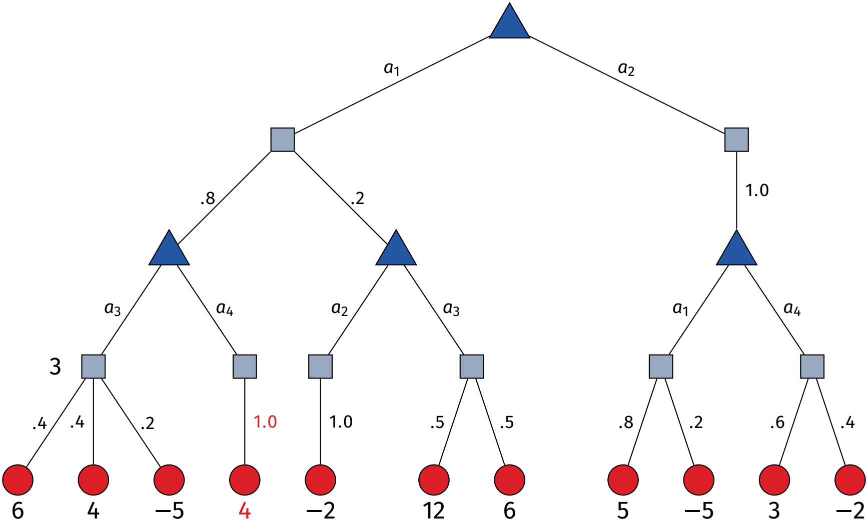
# Expectimax: Example



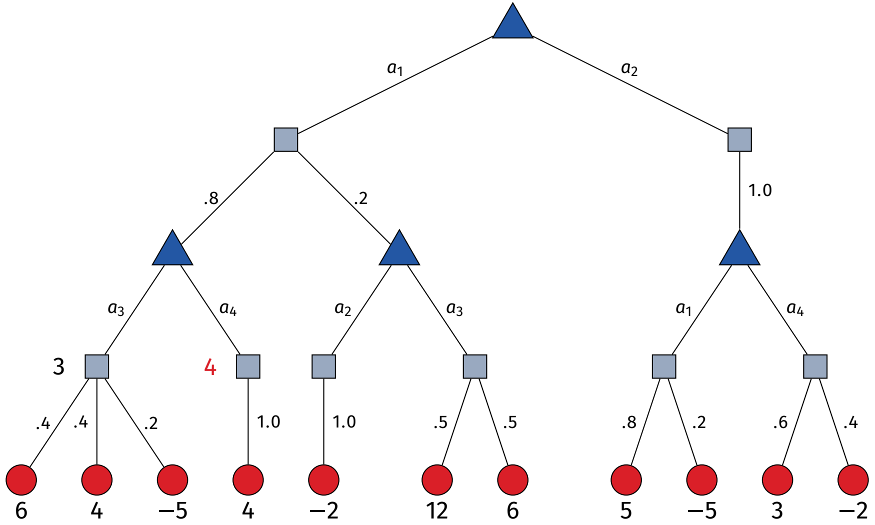
# Expectimax: Example



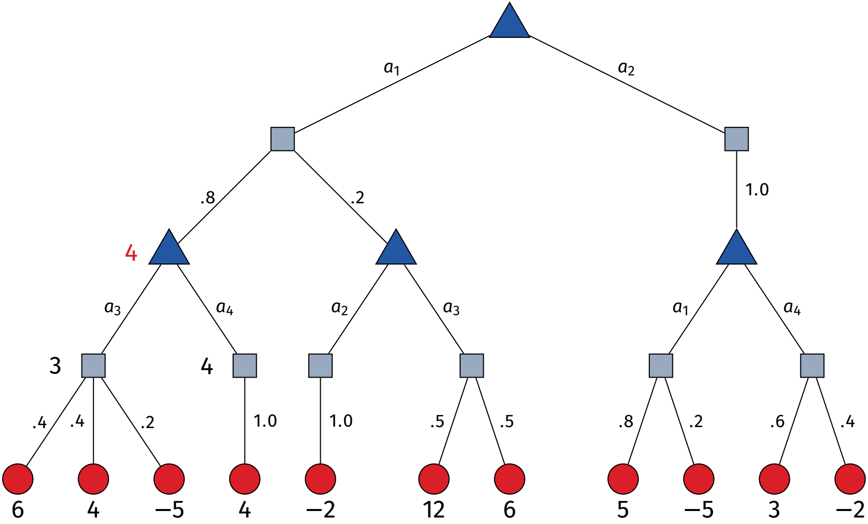
# Expectimax: Example



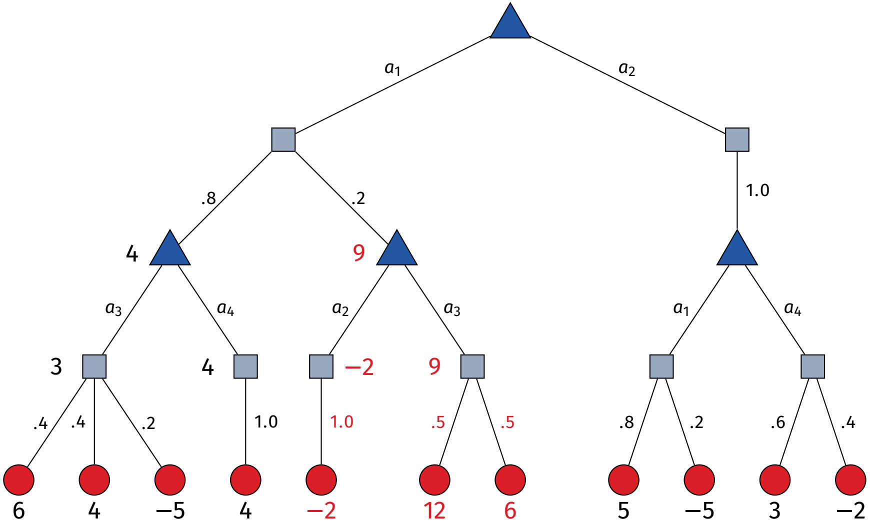
# Expectimax: Example



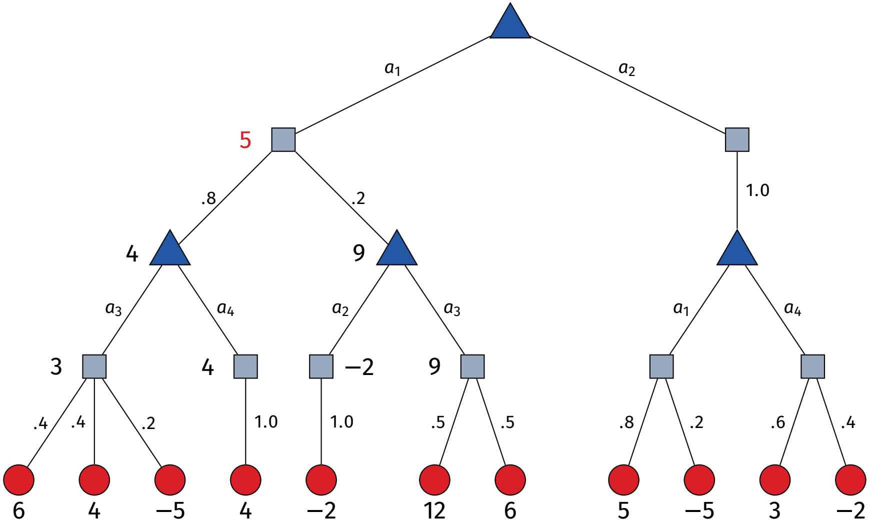
# Expectimax: Example



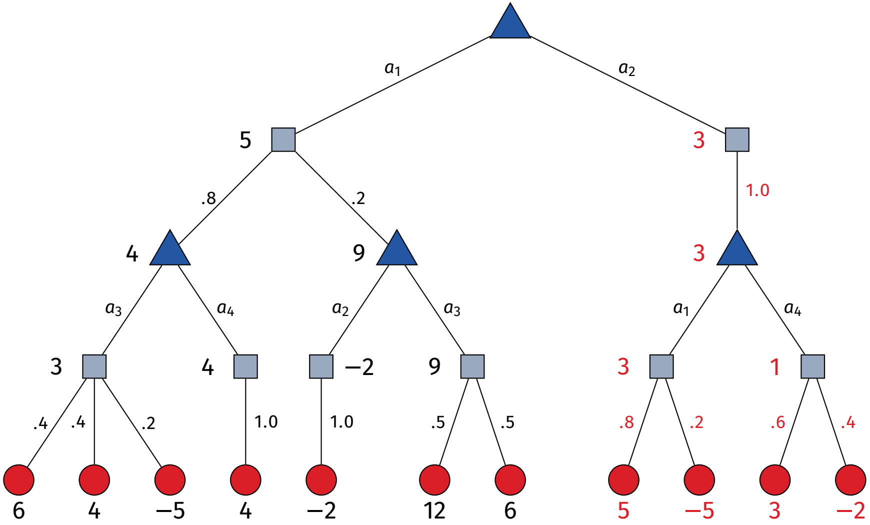
# Expectimax: Example



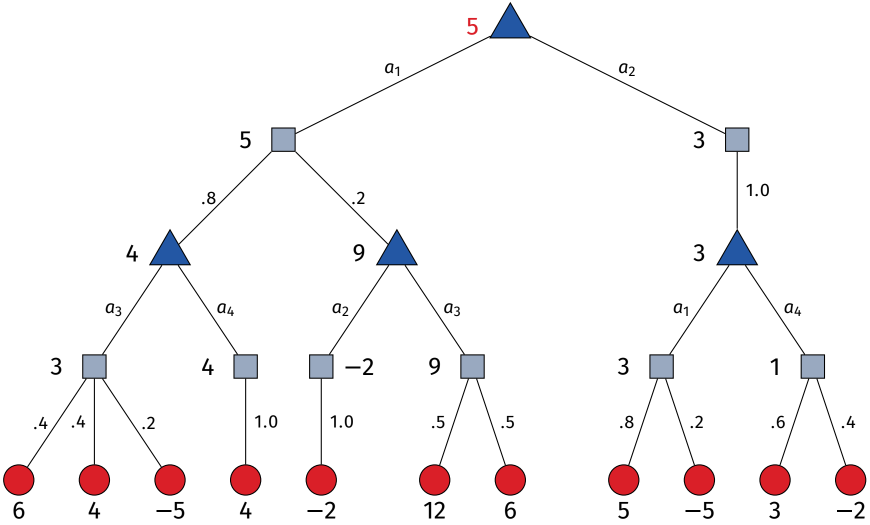
# Expectimax: Example



# Expectimax: Example



# Expectimax: Example



# Expectimax: Discussion

- expectimax computes (correct) expected utilities and action that yields the maximum expected utility
- on execution, the agent obtains exactly the utility value computed for the root in expectation
- but it can obtain a much higher or lower utility

# Expectimax: Discussion

- expectimax computes (correct) expected utilities and action that yields the maximum expected utility
- on execution, the agent obtains exactly the utility value computed for the root in expectation
- but it can obtain a much higher or lower utility

now we consider a more challenging environment:
Markov Decision Processes

# Markov Decision Processes

- Markov decision processes (MDPs) studied since the 1950s
- work up to 1980s mostly on theory and basic algorithms for small to medium sized MDPs
- today, focus on large, factored MDPs
- fundamental datastructure for probabilistic planning
- and for reinforcement learning
- different variants exist:
  - finite-horizon MDPs
  - stochastic shortest path problems
  - discounted reward infinite-horizon MDPs

# Stochastic Shortest Path Problems

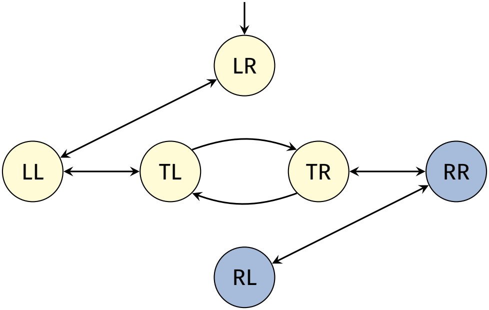
# Reminder: Transition Systems

## Definition (transition system)

A transition system (or search problem) is a 6-tuple $\mathcal{S} = \langle S, A, cost, T, s_I, S_\star \rangle$ with

- set of states $S$
- finite set of actions $A$
- action costs $cost : A \longrightarrow \mathbb{R}_0^+$
- transition model $T : S \times A \longrightarrow S \cup \{\bot\}$
- initial state $s_I \in S$
- set of goal states $S_\star \subseteq S$

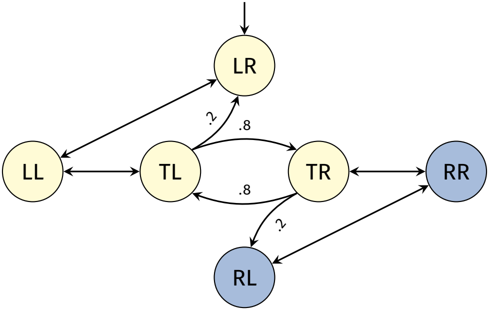
# Transition System Example




Logistics problem with one package, one truck, two locations:

- location of package: domain $\{L, R, T\}$
- location of truck: domain $\{L, R\}$

# Stochastic Shortest Path Example




Logistics problem with one package, one truck, two locations:

- location of package: $\{L, R, T\}$
- location of truck: $\{L, R\}$
- if truck moves with package, 20% chance of losing package

# Stochastic Shortest Path Problem

## Definition (Stochastic Shortest Path Problem)

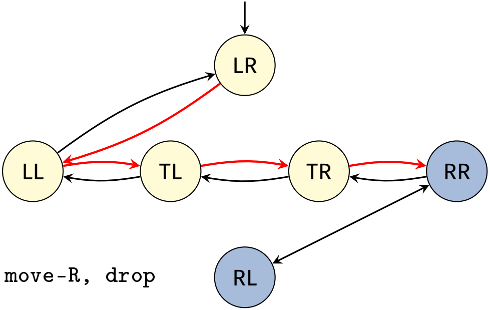
A stochastic shortest path problem (SSP) is a 6-tuple $\mathcal{T} = \langle S, A, c, T, s_I, S_\star \rangle$ with

- finite set of states $S$
- finite set of actions $A$
- action costs $cost : A \longrightarrow \mathbb{R}_0^+$
- transition function $T : S \times A \times S \mapsto [0, 1]$
- initial state $s_I \in S$
- set of goal states $S_\star \subseteq S$

For all $s \in S$ and $a \in A$ with $T(s, a, s') > 0$ for some $s' \in S$, we require $\sum_{s' \in S} T(s, a, s') = 1$.

We assume there is $a \in A$ and $s' \in S$ with $T(s, a, s') > 0$ for all $s \in S$.

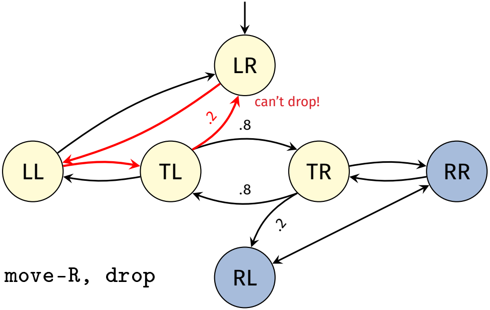
# Solutions in Transition Systems




`move-L, pickup, move-R, drop`

- in a deterministic transition system a solution is a plan, i.e., a sequence of operators that leads from $s_I$ to some $s_\star \in S_\star$
- an optimal solution is a cheapest possible plan
- a deterministic agent that executes a plan will reach the goal

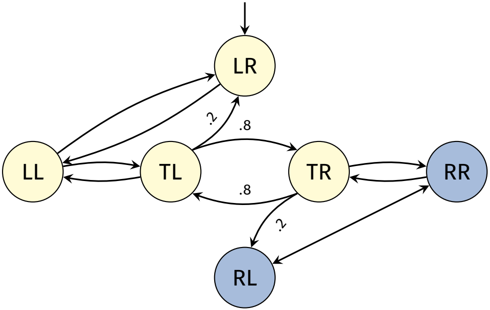
# Solutions in SSPs



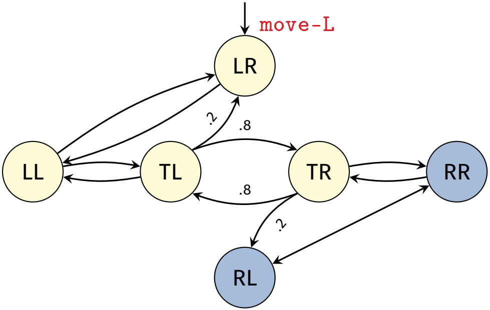

`move-L, pickup, move-R, drop`

- the same plan does not always work for the probabilistic agent (not reaching the goal or not being able to execute the plan)
- non-determinism can lead to a different outcome than anticipated in the plan
- need a policy

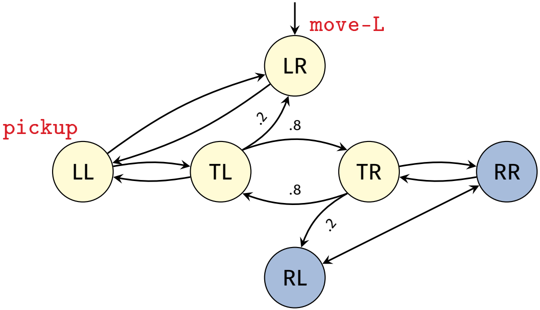
# Solutions in SSPs



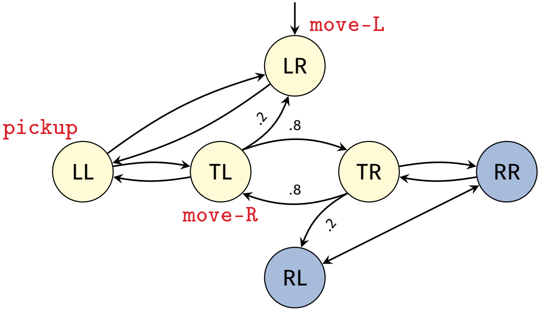
# Solutions in SSPs



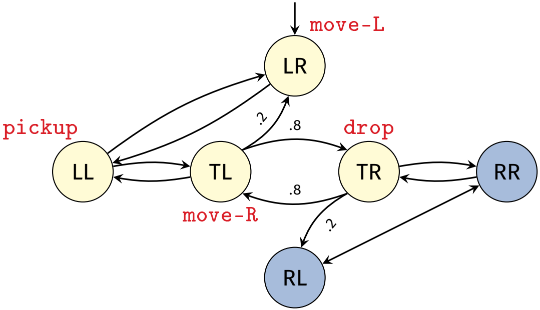
# Solutions in SSPs



# Solutions in SSPs



# Solutions in SSPs



# Policies for SSPs

## Definition (Policy for SSPs)

Let $\mathcal{T} = \langle S, A, c, T, s_I, S_\star \rangle$ be an SSP.
Let $\pi$ be a mapping $\pi : S \to A \cup \{\bot\}$ such that $\pi(s) \in A(s) \cup \{\bot\}$ for all $s \in S$.

The set of reachable states $S_\pi(s)$ from $s$ under $\pi$ is defined recursively as the smallest set satisfying the rules

- $s \in S_\pi(s)$ and
- $\mathrm{succ}(s', \pi(s')) \subseteq S_\pi(s)$ for all $s' \in S_\pi(s) \setminus S_\star$ where $\pi(s') \neq \bot$.

If $\pi(s') \neq \bot$ for all $s' \in S_\pi(s_I) \setminus S_\star$, then $\pi$ is a policy for $\mathcal{T}$.
If the probability to eventually reach a goal is 1 for all $s' \in S_\pi(s_I)$ then $\pi$ is a proper policy for $\mathcal{T}$.

# Additional Requirements for SSPs

- We make two requirements for SSPs:
  - There is a proper policy.
  - Every improper policy incurs infinite cost from every reachable state from which it does not reach a goal with probability 1.
- We only consider SSPs that satisfy these requirements.
- What does this mean in practice?
  - no unavoidable dead ends
  - no cost-free cyclic behavior possible
- With these requirements every cost-minimizing policy is a proper policy.

# Markov Decision Processes

# Discounted Reward Infinite-horizon MDPs

differences to initial example and SSPs:

- MDPs can be cyclic
- every action application yields a (positive or negative) reward (not only utilities in terminal states)
- aim is not to reach a goal state
- instead, agent acts forever (infinite horizon)
- aim: maximize expected overall reward
- earlier rewards count more than later rewards
  - rewards decay exponentially with discount factor $\gamma$:
    now full value $r$, in next step $\gamma r$, in two steps only $\gamma^2 r$, ...
  - ensures that algorithms converge
    despite cycles and infinite horizon

# Markov Decision Process

**Definition (Markov Decision Process)**

A (discounted reward) infinite-horizon Markov decision process (MDP) is a 6-tuple $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ with

- finite set of states $S$
- finite set of actions $A$
- reward function $R : S \times A \times S \rightarrow \mathbb{R}$
- transition function $T : S \times A \times S \mapsto [0, 1]$
- initial state $s_I \in S$
- discount factor $\gamma \in (0, 1)$

For all $s \in S$ and $a \in A$ with $T(s, a, s') > 0$ for some $s' \in S$,
we require $\sum_{s' \in S} T(s, a, s') = 1$.
We assume there is $a \in A$ and $s' \in S$ with $T(s, a, s') > 0$ for all $s \in S$.

# Markov Property

Markov decision processes are named after Russian mathematician Andrey Markov

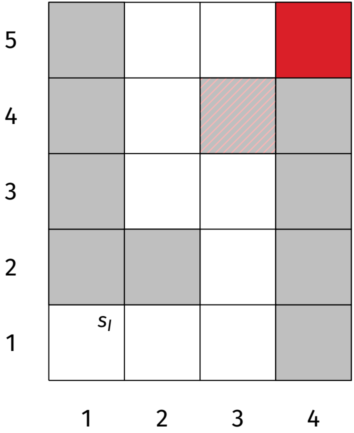
Markov property: immediate reward and probability distribution over successor states only depend on current state and applied action

⇝ not on previously visited states or earlier actions

# Some Terminology

- if $T(s, a, s') > 0$ for some $s'$, we say that $a$ is applicable in $s$
- the set of applicable actions in $s$ is $A(s)$
- the successor set of $s$ and $a$ is $\text{succ}(s, a) = \{s' \in S \mid T(s, a, s') > 0\}$

# MDP Example




- reward to `move` from striped cell is $-3$ ($-1$ everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

# Policy

## Definition (Policy)

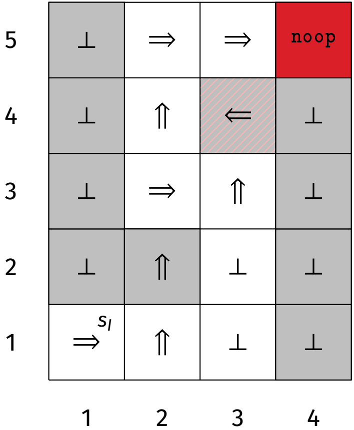
Let $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ be an MDP.
Let $\pi$ be a mapping $\pi : S \longrightarrow A \cup \{\bot\}$ such that $\pi(s) \in A(s) \cup \{\bot\}$ for all $s \in S$.

The set of reachable states $S_\pi(s)$ from $s$ under $\pi$ is defined recursively as the smallest set satisfying the rules

- $s \in S_\pi(s)$ and
- $\mathrm{succ}(s', \pi(s')) \subseteq S_\pi(s)$ for all $s' \in S_\pi(s)$ where $\pi(s') \neq \bot$.

If $\pi(s') \neq \bot$ for all $s' \in S_\pi(s_I)$, then $\pi$ is a policy for $\mathcal{T}$.

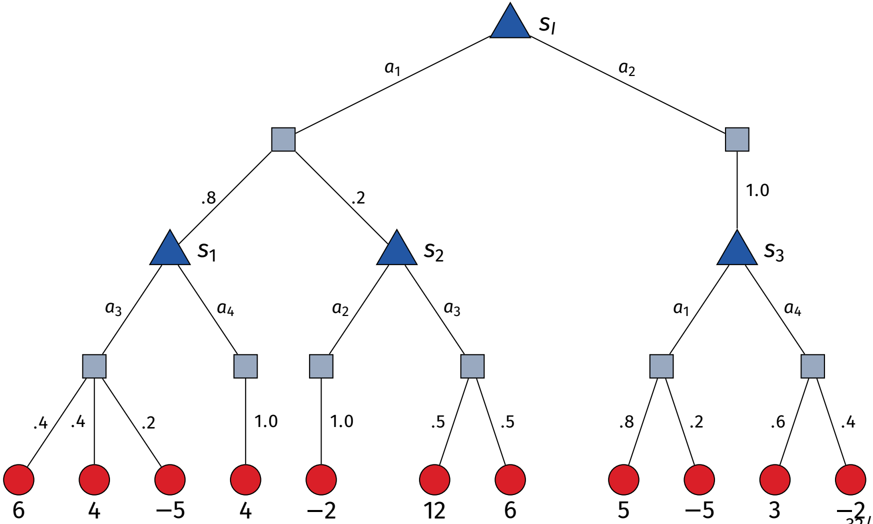
# Policy Example




- reward to `move` from striped cell is $-3$ ($-1$ everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

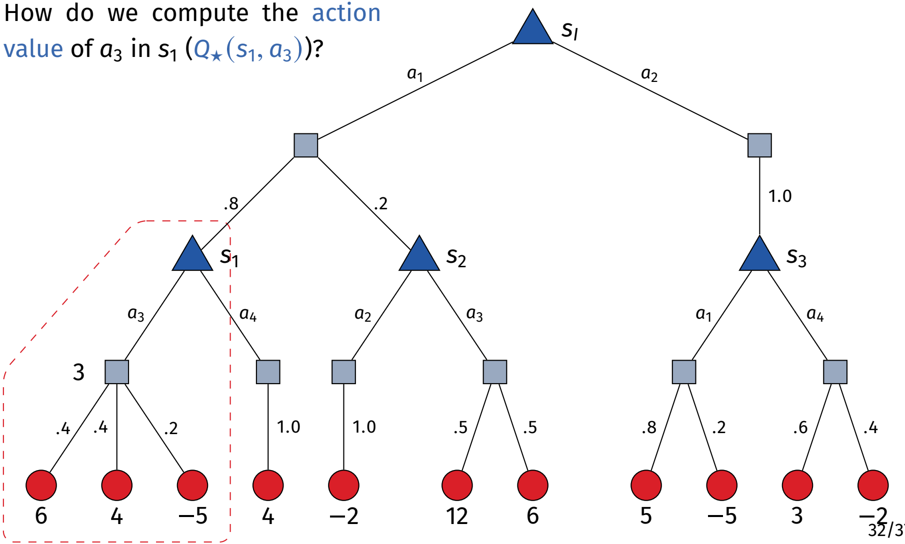
# Bellman Equation

# State and Action Value



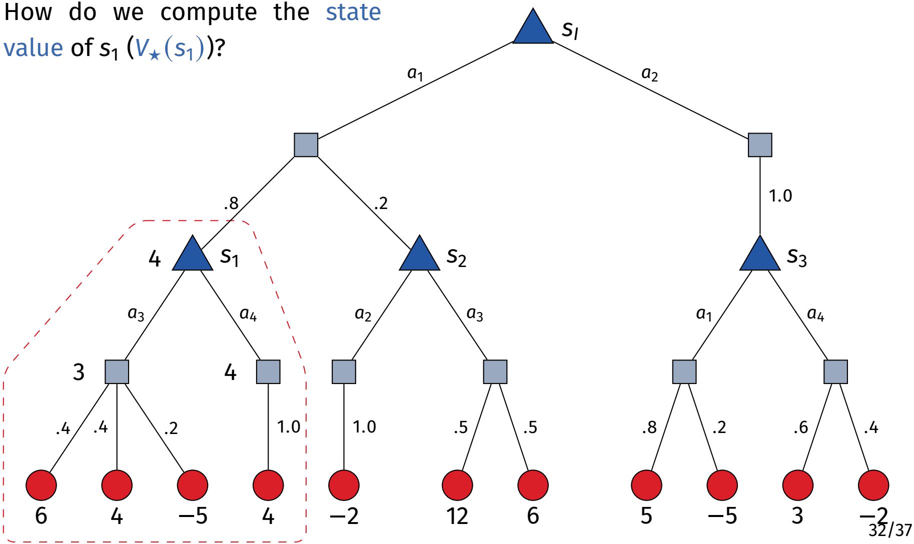
# State and Action Value

How do we compute the action value of $a_3$ in $s_1$ ($Q_\star(s_1, a_3)$)?



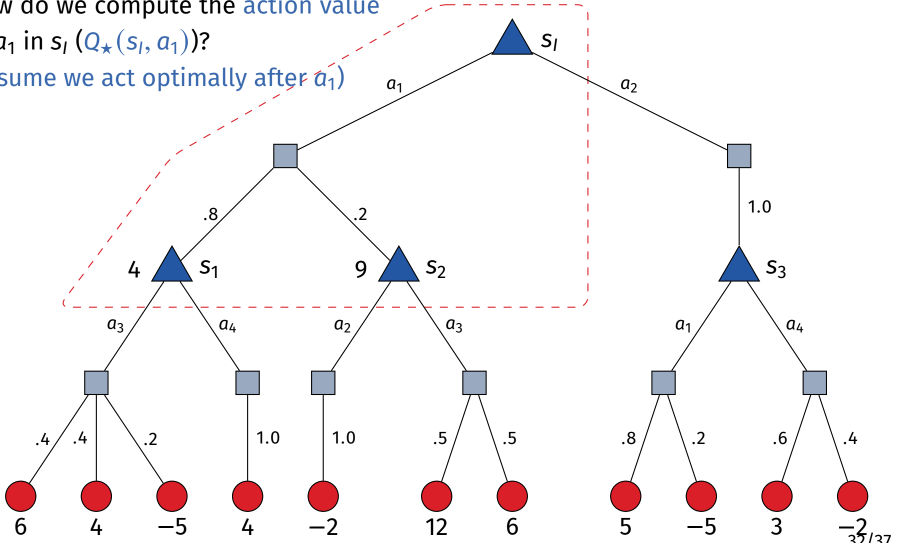
# State and Action Value

How do we compute the state value of $s_1$ ($V_\star(s_1)$)?



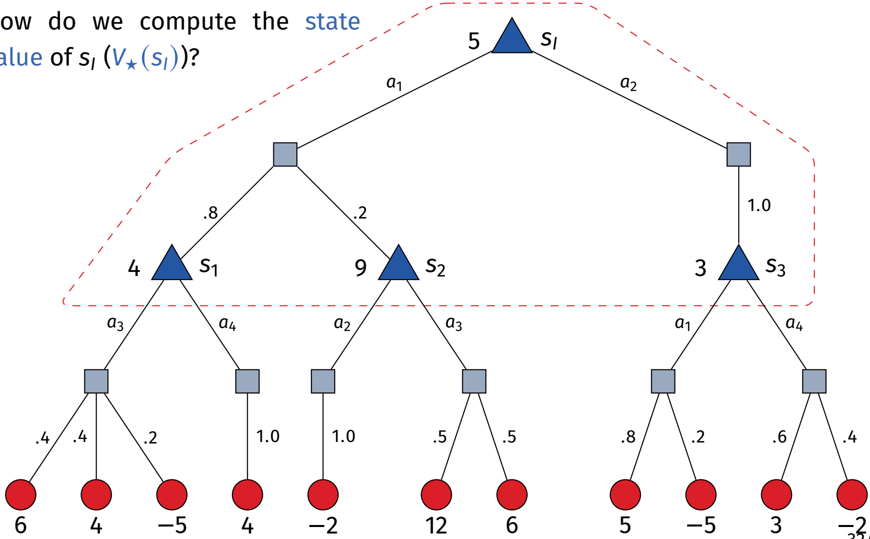
# State and Action Value

How do we compute the action value of $a_1$ in $s_I$ ($Q_\star(s_I, a_1)$)?
(assume we act optimally after $a_1$)



# State and Action Value

How do we compute the state value of $s_I$ ($V_\star(s_I)$)?



# Bellman Equation

Example not an MDP, but the idea also works in MDPs!

**Definition (Bellman Equation)**

Let $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ be an MDP.

The Bellman equation for a state $s$ of $\mathcal{T}$ is the set of equations that describes $V_\star(s)$, where

$$V_\star(s) := \max_{a \in A(s)} Q_\star(s, a)$$

$$Q_\star(s, a) := \sum_{s' \in \mathrm{succ}(s,a)} T(s, a, s') \cdot (R(s, a, s') + \gamma \cdot V_\star(s')) \,.$$

$V_\star(s)$ is the state-value of $s$ and

$Q_\star(s, a)$ is the action- or Q-value of $a$ in $s$.

# Optimal Policy

The solution $V_\star(s)$ of the Bellman equation describes the maximal expected reward that can be achieved from state $s$ in MDP $\mathcal{T}$.

What is the policy that achieves the maximal expected reward?

# Optimal Policy

The solution $V_\star(s)$ of the Bellman equation describes the maximal expected reward that can be achieved from state $s$ in MDP $\mathcal{T}$.

What is the policy that achieves the maximal expected reward?

**Definition (Optimal Policy)**

Let $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ be an MDP.
A policy $\pi$ is an optimal policy if $\pi(s) \in \arg\max_{a \in A(s)} Q_\star(s, a)$ for all $s \in S_\pi(s_I)$ and the expected reward of $\pi$ in $\mathcal{T}$ is $V_\star(s_I)$.

# Value Functions

**Definition (Value Functions)**

Let $\pi$ be a policy for MDP $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$.

The state-value $V_\pi(s)$ of $s \in S_\pi(s_I)$ under $\pi$ is defined as

$$V_\pi(s) := Q_\pi(s, \pi(s))$$

and the action- or Q-value $Q_\pi(s, a)$ of $s$ and $a$ under $\pi$ is defined as

$$Q_\pi(s, a) := \sum_{s' \in \mathrm{succ}(s,a)} T(s, a, s') \cdot (R(s, a, s') + \gamma \cdot V_\pi(s')) .$$

The state-value $V_\pi(s)$ describes the expected reward of applying $\pi$ in MDP $\mathcal{T}$, starting from $s$.

# Summary

# Summary

- SSPs are transition systems with a probabilistic transition relation.
- (Discounted-reward) MDPs allow state-dependent rewards that are discounted over an infinite horizon.
- Solutions of SSPs and MDPs are policies.
- For SSPs: minimize expected cost
- For MDPs: maximize expected reward
- The state-values of a policy specify the expected reward (cost) of following that policy.
- The Bellman equation describes the state-values of an optimal policy.