Planning and Optimization

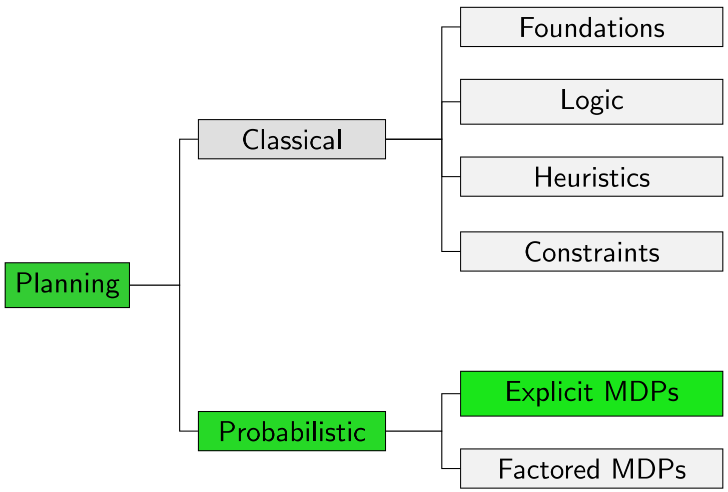
F1. Markov Decision Processes

Malte Helmert and Thomas Keller

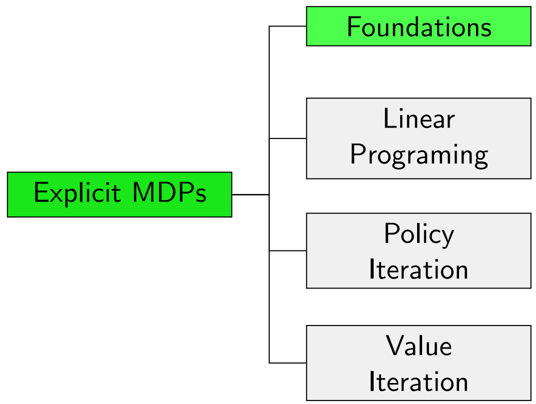
Universität Basel

November 27, 2019

Content of this Course

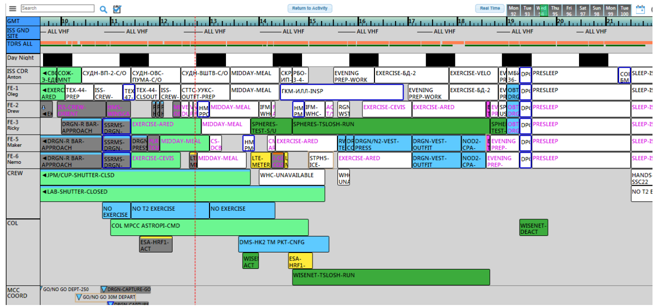


Content of this Course: Explicit MDPs



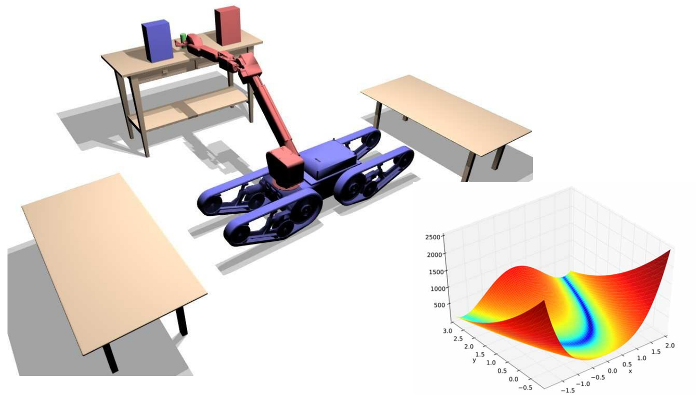
F1.1 Motivation

Generalization of Classical Planning: Temporal Planning



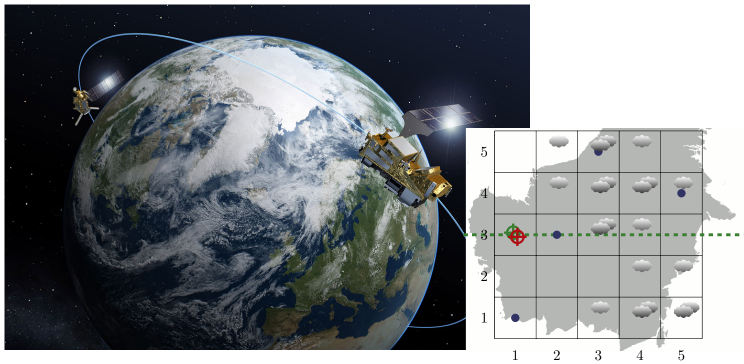
- timetable for astronauts on ISS
- concurrency required for some experiments
- optimize makespan

Generalization of Classical Planning: Numeric Planning



- kinematics of robotic arm
- state space is continuous
- preconditions and effects described by complex functions

Generalization of Classical Planning: MDPs

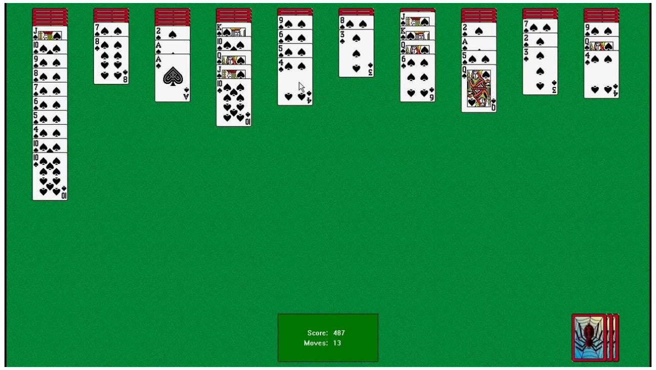


- satellite takes images of patches on earth
- weather forecast is uncertain
- find solution with lowest expected cost

Generalization of Classical Planning: Multiplayer Games

- Chess
- there is an opponent with a contradictory objective

Generalization of Classical Planning: POMDPs



- Solitaire
- some state information cannot be observed
- must reason over belief for good behaviour

Limitations of Classical Planning

- many applications are combinations of these
- all of these are active research areas
- we focus on one of them:
  probabilistic planning with Markov decision processes
- MDPs are closely related to games (Why?)

F1.2 Markov Decision Process

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 ($\rightsquigarrow$ Part F)
- Today, focus on large, factored MDPs ($\rightsquigarrow$ Part G)
- Fundamental datastructure for reinforcement learning (not covered in this course)
- and for probabilistic planning
- different variants exist

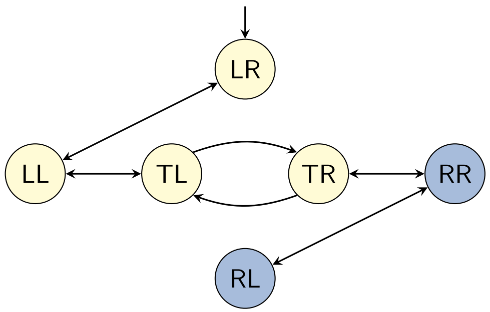
Reminder: Transition Systems

Definition (Transition System)

A transition system is a 6-tuple $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ where

- S is a finite set of states,
- L is a finite set of (transition) labels,
- $c : L \to \mathbb{R}_0^+$ is a label cost function,
- $T \subseteq S \times L \times S$ is the transition relation,
- $s_0 \in S$ is the initial state, and
- $S_\star \subseteq S$ is the set of goal states.

Reminder: Transition System Example



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

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

Stochastic Shortest Path Problem

Definition (Stochastic Shortest Path Problem)

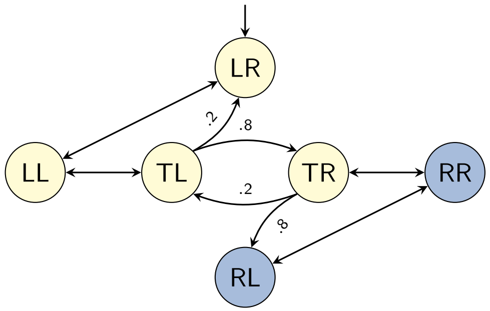
A stochastic shortest path problem (SSP) is a 6-tuple $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$, where

- S is a finite set of states,
- L is a finite set of (transition) labels (or actions),
- $c : L \to \mathbb{R}_0^+$ is a label cost function,
- $T : S \times L \times S \mapsto [0, 1]$ is the transition function,
- $s_0 \in S$ is the initial state, and
- $S_\star \subseteq S$ is the set of goal states.

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

Note: An SSP is the probabilistic pendant of a transition system.

Reminder: Transition System 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

Markov Decision Process

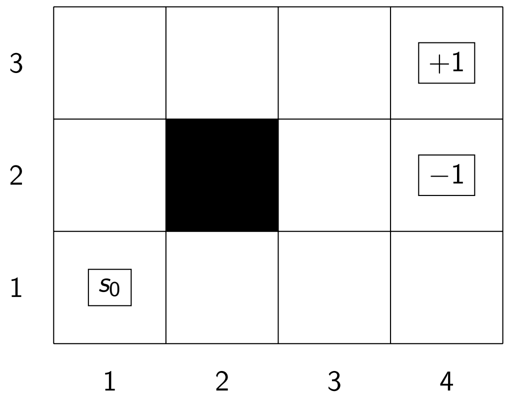
Definition (Markov Decision Process)

A (discounted reward) Markov decision process (MDP) is a 6-tuple $\mathcal{T} = \langle S, L, R, T, s_0, \gamma \rangle$, where

- S is a finite set of states,
- L is a finite set of (transition) labels (or actions),
- $R : S \times L \to \mathbb{R}$ is the reward function,
- $T : S \times L \times S \mapsto [0, 1]$ is the transition function,
- $s_0 \in S$ is the initial state, and
- $\gamma \in (0, 1)$ is the discount factor.

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

Example: Grid World



- moving *north* goes *east* with probability 0.4
- only applicable action in (4,2) and (4,3) is *collect*, which
  - sets position back to (1,1)
  - gives reward of +1 in (4,3)
  - gives reward of −1 in (4,2)

Terminology (1)

- If $p := T(s, \ell, s') > 0$, we write $s \xrightarrow{p:\ell} s'$ or $s \xrightarrow{p} s'$ if not interested in ℓ.
- If $T(s, \ell, s') = 1$, we also write $s \xrightarrow{\ell} s'$ or $s \to s'$ if not interested in ℓ.
- If $T(s, \ell, s') > 0$ for some s' we say that ℓ is applicable in s.
- The set of applicable actions in s is $L(s)$.
  We assume that $L(s) \neq \emptyset$ for all $s \in S$.

Terminology (2)

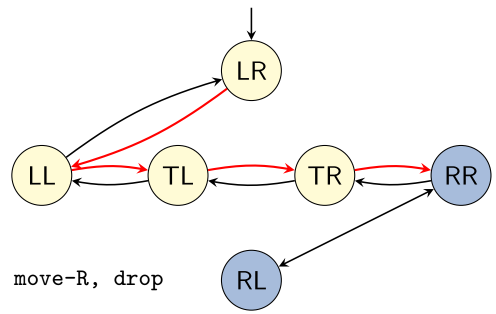
- the successor set of s and ℓ is $\text{succ}(s, \ell) = \{s' \in S \mid T(s, \ell, s') > 0\}$
- s' is a successor of s if $s' \in \text{succ}(s, \ell)$ for some ℓ
- with $s' \sim \text{succ}(s, \ell)$ we denote that successor $s' \in \text{succ}(s, \ell)$ of s and ℓ is sampled according to probability distribution T

Terminology (3)

- s' is reachable from s if there exists a sequence of transitions $s^0 \xrightarrow{p_1:\ell_1} s^1, \ldots, s^{n-1} \xrightarrow{p_n:\ell_n} s^n$ s.t. $s^0 = s$ and $s^n = s'$
  - Note: $n = 0$ possible; then $s = s'$
  - $s^0, \ldots, s^n$ is called (state) path from s to s'
  - $\ell_1, \ldots, \ell_n$ is called (action) path from s to s'
  - length of path is n
  - cost of path in SSP is $\sum_{i=1}^{n} c(\ell_i)$ and reward of path in MDP is $\sum_{i=1}^{n} \gamma^{i-1} R(s_{i-1}, \ell_i)$
  - s' is reached from s through this path with probability $\prod_{i=1}^{n} p_i$

F1.3 Policy

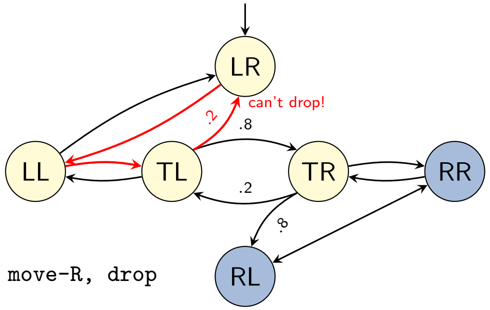
Solutions in SSPs



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

- solution in deterministic transition systems is plan, i.e., a goal path from s_0 to some $s_\star \in S_\star$
- cheapest plan is optimal solution
- deterministic agent that executes plan will reach goal

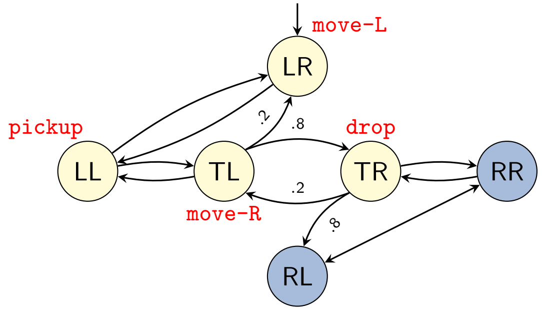
Solutions in SSPs



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

- probabilistic agent will not reach goal or cannot execute plan
- non-determinism can lead to different outcome than anticipated in plan
- require a more general solution: a policy

Solutions in SSPs



- policy must be allowed to be cyclic
- policy must be able to branch over outcomes
- policy assigns applicable actions to states

Policy for SSPs

Definition (Policy for SSPs)

Let $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ be an SSP. A policy for $\mathcal{T}$ is a mapping $\pi : S \to L \cup \{\bot\}$ such that $\pi(s) \in L(s) \cup \{\bot\}$ for all s.

The set of reachable states $S_\pi(s)$ from s under π 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 $s' \in S_\pi(s)$, then π is executable in s.

Policy Representation

- size of explicit representation of executable policy π is $|S_\pi(s_0)|$
- often, $|S_\pi(s_0)|$ similar to $|S|$
- compact policy representation, e.g. via value function approximation or neural networks, is active research area
  $\Rightarrow$ not covered in this course
- instead, we consider small state spaces for basic algorithms
- or online planning where planning for the current state s_0 is interleaved with execution of $\pi(s_0)$

Policy for MDPs

Definition (Policy for MDPs)

Let $\mathcal{T} = \langle S, L, R, T, s_0, \gamma \rangle$ be an MDP. A policy for $\mathcal{T}$ is a mapping $\pi : S \to L \cup \{\perp\}$ such that $\pi(s) \in L(s) \cup \{\perp\}$ for all s.

The set of reachable states $S_\pi(s)$ from s under π 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 $\pi(s') \neq \perp$.

If $\pi(s') \neq \perp$ for all $s' \in S_\pi(s)$, then π is executable in s.

F1.4 Summary

Summary

- Many planning scenarios beyond classical planning
- Part F and G are on probabilistic planning
- SSPs are classical planning + probabilistic transition function
- MDPs allow state-dependent rewards that are discounted over an infinite horizon
- Solutions of SSPs and MDPs are policies
- Policies consider branching and cycles