

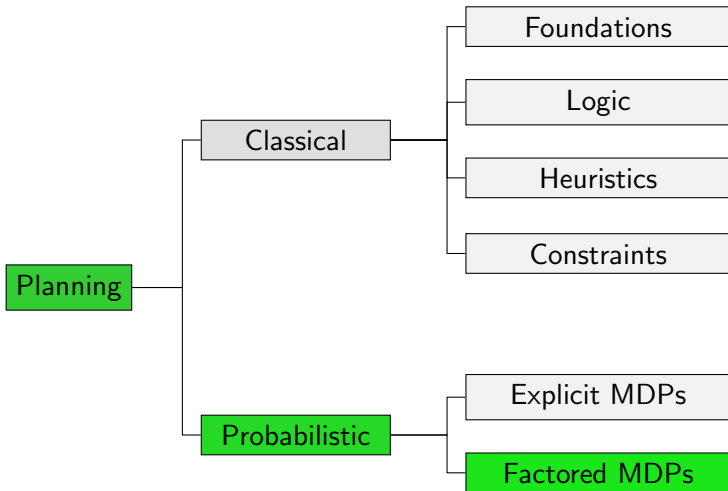
# Planning and Optimization

## G1. Factored MDPs

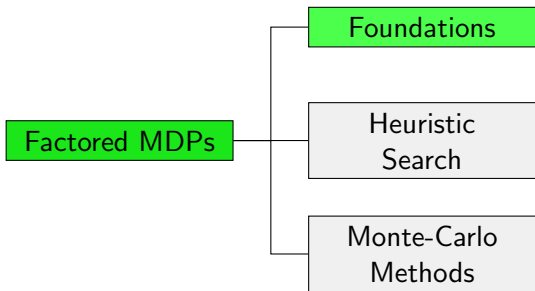
Malte Helmert and Gabriele Röger

Universität Basel

# Content of this Course



# Content of this Course: Factored MDPs



# Factored MDPs

# Factored MDPs

We would like to specify MDPs and SSPs with large state spaces. In classical planning, we introduced **planning tasks** to represent large transition systems compactly.

- represent aspects of the world in terms of **state variables**
- states are a **valuation of state variables**
- $n$  (propositional) state variables induce  $2^n$  states  
     $\rightsquigarrow$  **exponentially more compact** than “explicit” representation

# Finite-Domain State Variables

## Definition (Finite-Domain State Variable)

A **finite-domain state variable** is a symbol  $v$  with an associated **domain**  $\text{dom}(v)$ , which is a finite non-empty set of values.

Let  $V$  be a finite set of finite-domain state variables.

A **state**  $s$  over  $V$  is an assignment  $s : V \rightarrow \bigcup_{v \in V} \text{dom}(v)$  such that  $s(v) \in \text{dom}(v)$  for all  $v \in V$ .

A **formula** over  $V$  is a propositional logic formula whose atomic propositions are of the form  $v = d$  where  $v \in V$  and  $d \in \text{dom}(v)$ .

For simplicity, we only consider finite-domain state variables here.

# Syntax of Operators

## Definition (SSP and MDP Operators)

An **SSP operator**  $o$  over a set of state variables  $V$  has three components:

- a **precondition**  $pre(o)$ , a logical formula over  $V$
- an **effect**  $eff(o)$  over  $V$ , defined on the following slides
- a **cost**  $cost(o) \in \mathbb{R}_0^+$

An **MDP operator**  $o$  over a set of state variables  $V$  has three components:

- a **precondition**  $pre(o)$ , a logical formula over  $V$
- an **effect**  $eff(o)$  over  $V$ , defined on the following slides
- a **reward**  $reward(o)$  over  $V$ , defined on the following slides

Whenever we just say **operator** (without SSP or MDP), both kinds of operators are allowed.

# Syntax of Effects

## Definition (Effect)

**Effects** over state variables  $V$  are inductively defined as follows:

- If  $v \in V$  is a finite-domain state variable and  $d \in \text{dom}(v)$ , then  $v := d$  is an effect (**atomic effect**).
- If  $e_1, \dots, e_n$  are effects, then  $(e_1 \wedge \dots \wedge e_n)$  is an effect (**conjunctive effect**).  
The special case with  $n = 0$  is the **empty effect**  $\top$ .
- If  $e_1, \dots, e_n$  are effects and  $p_1, \dots, p_n \in [0, 1]$  such that  $\sum_{i=1}^n p_i = 1$ , then  $(p_1 : e_1 \mid \dots \mid p_n : e_n)$  is an effect (**probabilistic effect**).

**Note:** To simplify definitions, conditional effects are omitted.



# Effects: Intuition

## Intuition for effects:

- **Atomic effects** can be understood as assignments that update the value of a state variable.
- A **conjunctive effect**  $e = (e_1 \wedge \dots \wedge e_n)$  means that all subeffects  $e_1, \dots, e_n$  take place simultaneously.
- A **probabilistic effect**  $e = (p_1 : e_1 | \dots | p_n : e_n)$  means that exactly one subeffect  $e_i \in \{e_1, \dots, e_n\}$  takes place with probability  $p_i$ .

# Semantics of Effects

## Definition

The **effect set**  $[e]$  of an effect  $e$  is a set of pairs  $\langle p, w \rangle$ , where  $p$  is a probability  $0 < p \leq 1$  and  $w$  is a partial assignment. The effect set  $[e]$  is the set obtained recursively as

$$[v := d] = \{\langle 1.0, \{v \mapsto d\} \rangle\},$$

$$[e \wedge e'] = \biguplus_{\langle p, w \rangle \in [e], \langle p', w' \rangle \in [e']} \{\langle p \cdot p', w \cup w' \rangle\},$$

$$[p_1 : e_1 \mid \dots \mid p_n : e_n] = \biguplus_{i=1}^n \{\langle p_i \cdot p, w \rangle \mid \langle p, w \rangle \in [e_i]\}.$$

where  $\biguplus$  is like  $\bigcup$  but merges  $\langle p, w' \rangle$  and  $\langle p', w' \rangle$  to  $\langle p + p', w' \rangle$ .

# Semantics of Operators

## Definition (Applicable, Outcomes)

Let  $V$  be a set of finite-domain state variables.

Let  $s$  be a state over  $V$ , and let  $o$  be an operator over  $V$ .

Operator  $o$  is **applicable** in  $s$  if  $s \models \text{pre}(o)$ .

The **outcomes** of applying an operator  $o$  in  $s$ , written  $s[o]$ , are

$$s[o] = \bigsqcup_{\langle p, w \rangle \in [\text{eff}(o)]} \{ \langle p, s'_w \rangle \},$$

with  $s'_w(v) = d$  if  $v = d \in w$  and  $s'_w(v) = s(v)$  otherwise  
and  $\bigsqcup$  is like  $\cup$  but merges  $\langle p, s' \rangle$  and  $\langle p', s' \rangle$  to  $\langle p + p', s' \rangle$ .

# Rewards

## Definition (Reward)

A **reward** over state variables  $V$  is inductively defined as follows:

- $c \in \mathbb{R}$  is a reward
- If  $\chi$  is a propositional formula over  $V$ ,  $[\chi]$  is a reward
- If  $r$  and  $r'$  are rewards,  $r + r'$ ,  $r - r'$ ,  $r \cdot r'$  and  $\frac{r}{r'}$  are rewards

Applying an MDP operator  $o$  in  $s$  **induces reward**  $\text{reward}(o)(s)$ , i.e., the value of the arithmetic function  $\text{reward}(o)$  where all occurrences of  $v \in V$  are replaced with  $s(v)$ .

# Probabilistic Planning Tasks

# Probabilistic Planning Tasks

## Definition (SSP and MDP Planning Task)

An **SSP planning task** is a 4-tuple  $\Pi = \langle V, I, O, \gamma \rangle$  where

- $V$  is a finite set of **finite-domain state variables**,
- $I$  is a valuation over  $V$  called the **initial state**,
- $O$  is a finite set of **SSP operators** over  $V$ , and
- $\gamma$  is a formula over  $V$  called the **goal**.

An **MDP planning task** is a 4-tuple  $\Pi = \langle V, I, O, d \rangle$  where

- $V$  is a finite set of **finite-domain state variables**,
- $I$  is a valuation over  $V$  called the **initial state**,
- $O$  is a finite set of **MDP operators** over  $V$ , and
- $d \in (0, 1)$  is the **discount factor**.

A **probabilistic planning task** is an SSP or MDP planning task.

# Mapping SSP Planning Tasks to SSPs

## Definition (SSP Induced by an SSP Planning Task)

The SSP planning task  $\Pi = \langle V, I, O, \gamma \rangle$  **induces** the SSP  $\mathcal{T} = \langle S, A, c, T, s_0, S_\star \rangle$ , where

- $S$  is the set of all states over  $V$ ,
- $A$  is the set of operators  $O$ ,
- $c(o) = \text{cost}(o)$  for all  $o \in O$ ,
- $T(s, o, s') = \begin{cases} p & \text{if } o \text{ applicable in } s \text{ and } \langle p, s' \rangle \in s[[o]] \\ 0 & \text{otherwise} \end{cases}$
- $s_0 = I$ , and
- $S_\star = \{s \in S \mid s \models \gamma\}$ .

# Mapping MDP Planning Tasks to MDPs

## Definition (MDP Induced by an MDP Planning Task)

The MDP planning task  $\Pi = \langle V, I, O, d \rangle$  **induces** the MDP  $\mathcal{T} = \langle S, A, R, T, s_0, \gamma \rangle$ , where

- $S$  is the set of all states over  $V$ ,
- $A$  is the set of operators  $O$ ,
- $R(s, o) = \text{reward}(o)(s)$  for all  $o \in O$  and  $s \in S$ ,
- $T(s, o, s') = \begin{cases} p & \text{if } o \text{ applicable in } s \text{ and } \langle p, s' \rangle \in s[[o]] \\ 0 & \text{otherwise} \end{cases}$
- $s_0 = I$ , and
- $\gamma = d$ .



# Complexity

# Complexity of Probabilistic Planning

## Definition (Policy Existence)

**Policy existence (POLICYEX)** is the following decision problem:

**GIVEN:** SSP planning task  $\Pi$

**QUESTION:** Is there a proper policy for  $\Pi$ ?

# Membership in EXP

## Theorem

$\text{POLICYEX} \in \text{EXP}$

## Proof.

The number of states in an SSP planning task is exponential in the number of variables. The induced SSP can be solved in time polynomial in  $|S| \cdot |A|$  via linear programming and hence in time exponential in the input size. □

# EXP-completeness of Probabilistic Planning

## Theorem

*POLICYEX is EXP-complete.*

## Proof Sketch.

Membership for POLICYEX: see previous slide.

Hardness is shown by Littman (1997) by reducing the EXP-complete game  $G_4$  to POLICYEX.

# Estimated Policy Evaluation

# Large SSPs and MDPs

- Before: optimal policies and exact state-values for small SSPs and MDPs.
- Now: focus on large SSPs and MDPs
- Further algorithms not necessarily optimal (may generate suboptimal policies)

# Interleaved Planning & Execution

- Number of reachable states of a policy usually **exponential** in the number of state variables
- For large SSPs and MDPs, policies cannot be provided **explicitly**.
- **Solution**: (possibly approximate) **compact representation** of policy required to describe solution  
⇒ not part of this lecture.
- **Alternative solution**: interleave planning and execution

# Interleaved Planning & Execution for SSPs

Plan-execute-monitor cycle for SSP  $\mathcal{T}$ :

- plan action  $a$  for the current state  $s$
- execute  $a$
- observe new current state  $s'$
- set  $s := s'$
- repeat until  $s \in \mathcal{S}_*$



# Interleaved Planning & Execution for MDPs

Plan-execute-monitor cycle for MDP  $\mathcal{T}$ :

- plan action  $a$  for the current state  $s$
- execute  $a$
- observe new current state  $s'$
- set  $s := s'$
- repeat until **discounted reward sufficiently small**

## Interleaved Planning & Execution in Practice

- avoids **loss of precision** that often comes with compact description of policy
- does not waste time with planning for states that are **never reached** during execution
- **poor decisions** can be avoided by spending more time with planning before execution
- in SSPs, this can even mean that computed policy is **not proper** and execution never reaches the goal
- in MDPs, it is not clear when the **discounted reward is sufficiently small**

# Estimated Policy Evaluation

- The **quality** of a policy is described by the state-value of the initial state  $V_\pi(s_0)$
  - Quality of given policy  $\pi$  can be computed (via **LP** or **backward induction**) or approximated arbitrarily closely (via **iterative policy evaluation**) in small SSPs or MDPs
  - **Impossible** if planning and execution are interleaved as policy is **incomplete**
- ⇒ **Estimate** quality of policy  $\pi$  by **executing** it  $n \in \mathbb{N}$  times

## Executing a Policy

### Definition (Run in SSP)

Let  $\mathcal{T}$  be an SSP and  $\pi$  be a proper policy for  $\mathcal{T}$ .

A sequence of transitions

$$\rho_\pi = s_0 \xrightarrow{p_1:\pi(s_0)} s_1, \dots, s_{n-1} \xrightarrow{p_n:\pi(s_{n-1})} s_n$$

is a **run**  $\rho_\pi$  of  $\pi$  if  $s_{i+1} \sim s_i[\pi(s_i)]$  and  $s_n \in S_\star$ .

The **cost** of run  $\rho_\pi$  is  $cost(\rho_\pi) = \sum_{i=0}^{n-1} cost(\pi(s_i))$ .

A run in an SSP can easily be generated by executing  $\pi$  from  $s_0$  until a state  $s \in S_\star$  is encountered.

## Executing a Policy

### Definition (Run in MDP)

Let  $\mathcal{T}$  be an MDP and  $\pi$  be a policy for  $\mathcal{T}$ .

A sequence of transitions

$$\rho_\pi = s_0 \xrightarrow{p_1:\pi(s_0)} s_1, \dots, s_{n-1} \xrightarrow{p_n:\pi(s_{n-1})} s_n$$

is a **run**  $\rho_\pi$  of  $\pi$  if  $s_{i+1} \sim s_i[\pi(s_i)]$ .

The **reward** of run  $\rho_\pi$  is  $reward(\rho_\pi) = \sum_{i=0}^{n-1} \gamma^i \cdot reward(s_i, \pi(s_i))$ .

To generate a run, a termination criterion (e.g., based on the change of the accumulated reward) must be specified.

# Estimated Policy Evaluation

## Definition (Estimated Policy Evaluation)

Let  $\mathcal{T}$  be an SSP,  $\pi$  be a policy for  $\mathcal{T}$  and  $\langle \rho_\pi^1, \dots, \rho_\pi^n \rangle$  be a sequence of runs of  $\pi$ .

The **estimated quality** of  $\pi$  via **estimated policy evaluation** is

$$\tilde{V}_\pi := \frac{1}{n} \cdot \sum_{i=1}^n \text{cost}(\rho_\pi^i).$$

# Convergence of Estimated Policy Evaluation in SSPs

## Theorem

Let  $\mathcal{T}$  be an SSP,  $\pi$  be a policy for  $\mathcal{T}$  and  $\langle \rho_\pi^1, \dots, \rho_\pi^n \rangle$  be a sequence of runs of  $\pi$ .

Then  $\tilde{V}_\pi \rightarrow V_\pi(s_0)$  for  $n \rightarrow \infty$ .

## Proof.

Holds due to the **strong law of large numbers**. □

$\Rightarrow \tilde{V}_\pi$  is a **good approximation** of  $v_\pi(s_0)$  if  $n$  sufficiently large.

# Summary



# Summary

- MDP and SSP planning tasks represent MDPs and SSPs **compactly**.
- Policy existence in SSPs is **EXP-complete**.
- **Interleaving planning and execution** avoids representation issues of (typically exponentially sized) policy.
- Quality of such an incomplete policy can be **estimated** by executing it a fixed number of times.
- In SSPs, **estimated policy evaluation** converges to the true quality of the policy.