Asociación Española para la Inteligencia Artificial (**AEPIA**)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Planificación Automática y Aprendizaje por Refuerzo para problemas de toma de decisión

Escuela de Verano de Inteligencia Artificial

Eva Onaindía
Valencian Research Artificial Intelligence Institute (VRAIN)
Universitat Politècnica de València

14 Junio 2023

Syllabus

1. Decision-making problems (decision-making control problems)
2. Reasoning to solve a control problem
3. Learning to solve a control problem

Decision-making control problem

A control problem involves a system that is described by a set of state variables

The system is controlled by making decisions that change the state variables

Decision: a course of action purposely chosen along time to achieve some goals

An agent with acting capabilities make control decisions about which action to execute in a state of the problem that changes the state variables

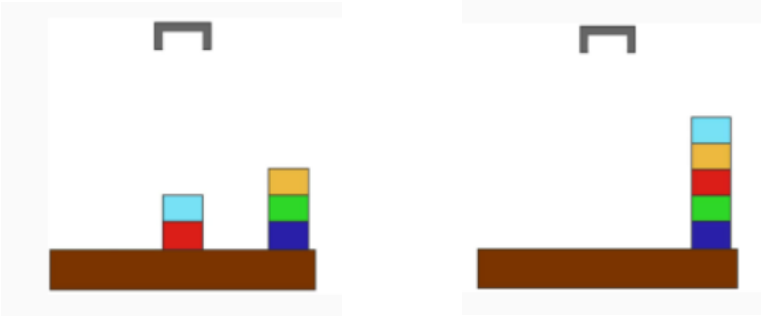
Goals:

- attainment goal
- maintenance goal
- maximize score

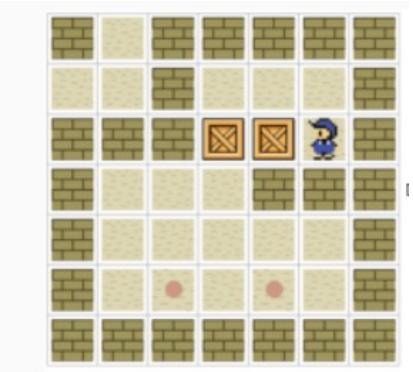
Actions:

- movement actions
- delivery operations
- performance actions
- …

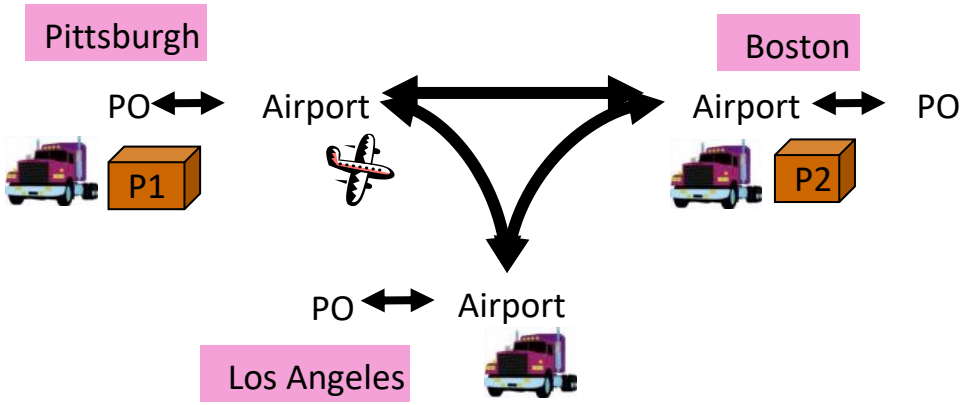
Decision-making control problem



Blocksworld domain

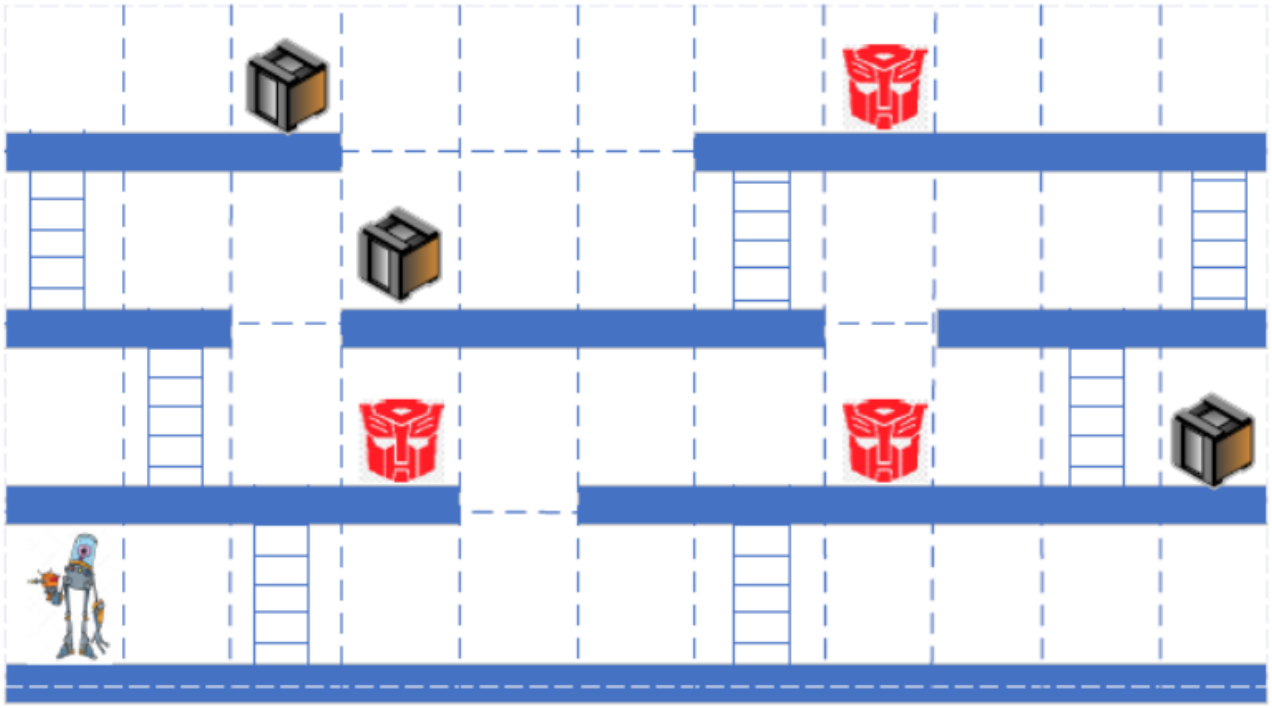


Sokoban



Logistics domain

Decision-making control problem

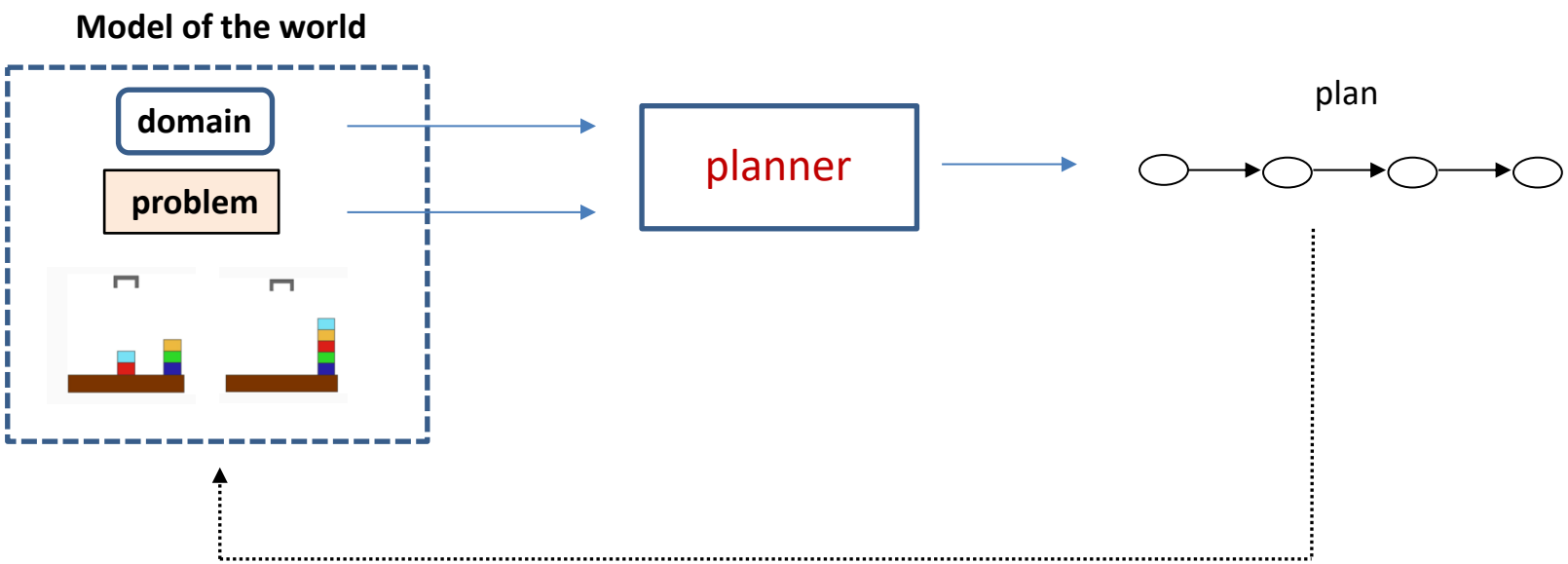


Robot-Building domain

Reasoning to solve a control problem

- Design a plan to complete the task before acting = offline planning
- A model specification of the world is needed → world model
  - state variables
  - action model
  - initial state
  - goal state

  (state variables + action model: **domain**; initial state + goal state: **problem**)
- Control is synthesized from the model (problem specification)
- A solver (planner) finds a solution plan for the problem (controller)



Reasoning to solve a control problem

- Design a plan before acting = offline planning
- A model specification is needed

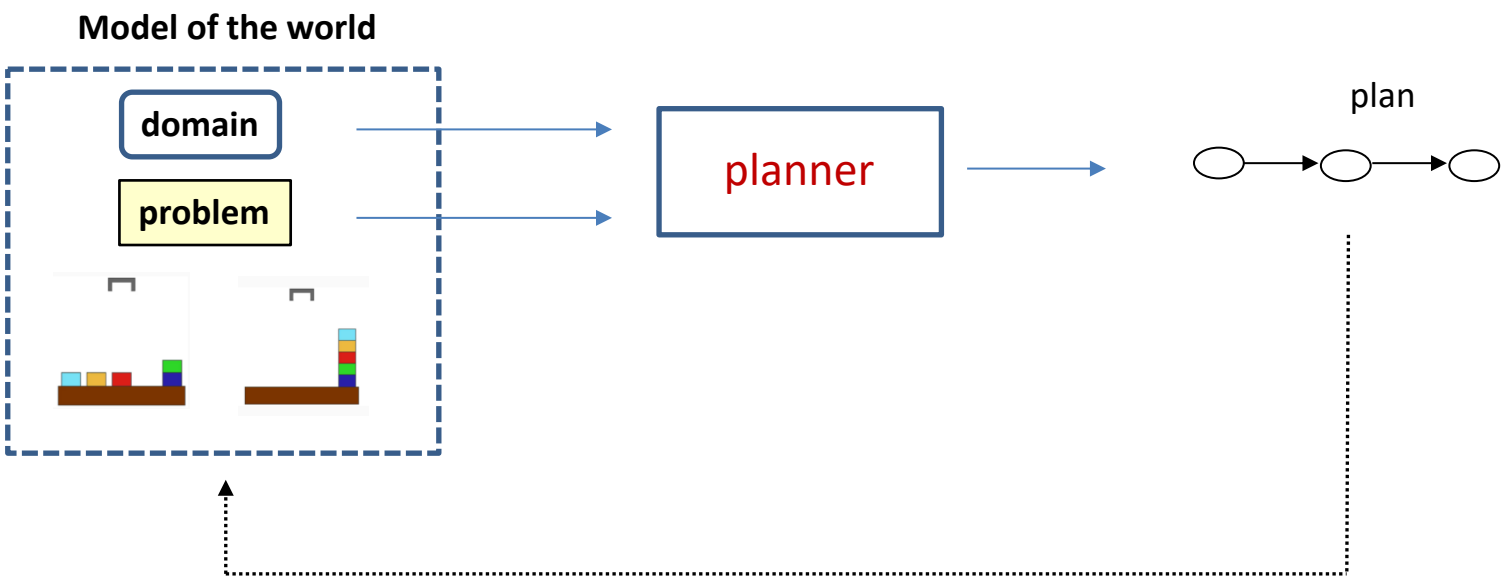
world model
- state variables
- action model

domain

- initial state
- goal state

problem

- Control is synthesized from the model (problem specification)
- A solver (panner) finds a solution plan for the problem (controller)



Reasoning to solve a control problem

- Design a plan before acting = offline planning
- A model specification is needed

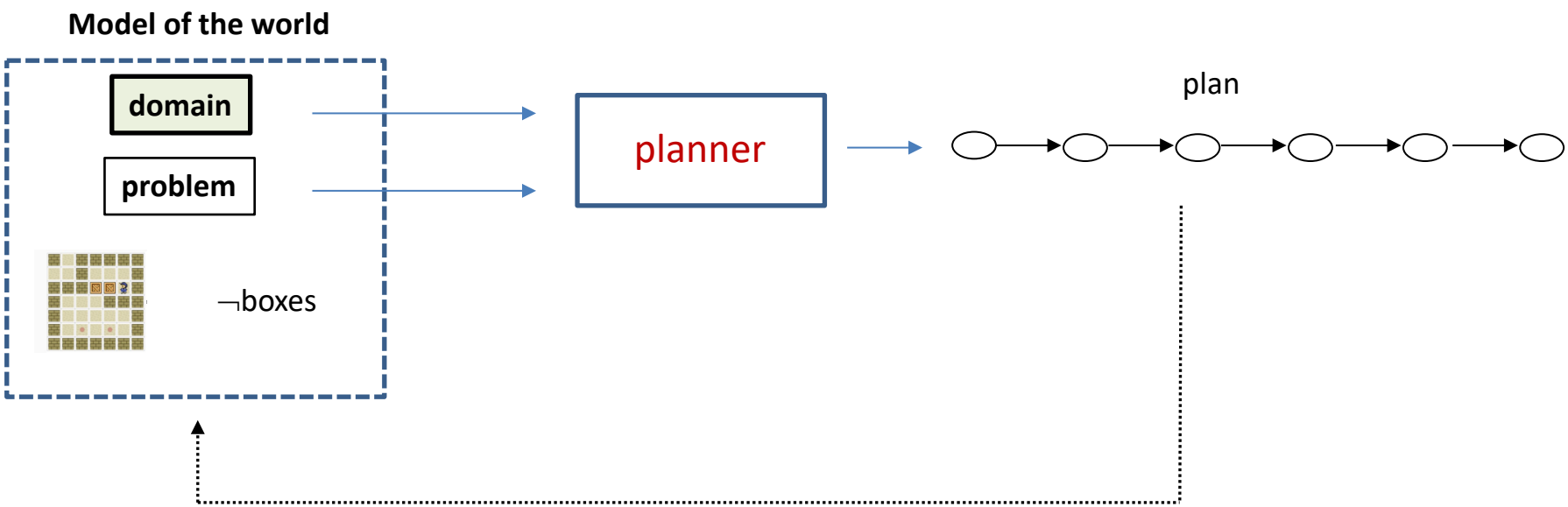
world model
- state variables
- action model

domain

- initial state
- goal state

problem

- Control is synthesized from the model (problem specification)
- A solver (planner) finds a solution plan for the problem (controller)



Reasoning to solve a control problem

Model of the world	Solver (planner)
representation language	search problem (state space, plan space, …)
factored representation (state variables)	heuristic-based search
actions (deterministic, non-deterministic, probabilistic)	SAT-based planning
observability (full, partial)	CSP-based planning
agency (single, multiple)	SMT-based planning

Reasoning to solve a control problem: model representation

Planning task: $P = \langle V, A, I, G \rangle$

domain

- V: state variables
- A: actions

problem

- I: initial state
- G: goal state

Planning Domain Description Language (PDDL):

- popularized by the International Planning Competitions (IPC)
- logic-based language → V: Boolean variables
- object-type hierarchy
- predicates:

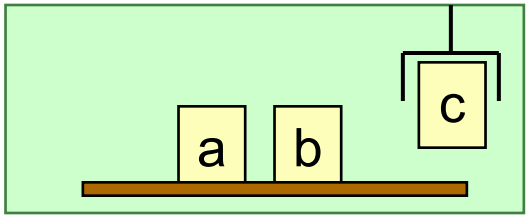
  (on ?x – block ?y – block)
  (holding ?x – block)
  (in ?x – package ?y – truck)

Reasoning to solve a control problem: model representation

domain

```
 (:predicates
        (on ?x - block ?y - block)
        (ontable ?x - block)
        (clear ?x - block)
        (armempty)
        (holding ?x - block)
  )
```

```
(:action stack
  :parameters (?ob - block ?underob - block)
  :precondition
      (and
        (clear ?underob)
        (holding ?ob))
  :effect
      (and
        (clear ?ob)
        (arm-empty)
        (on ?ob ?underob)
        (not (holding ?ob))
        (not (clear ?underob))
      )
)
```

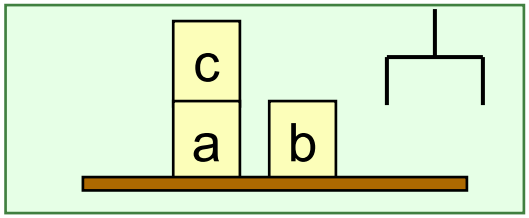


Reasoning to solve a control problem: model representation

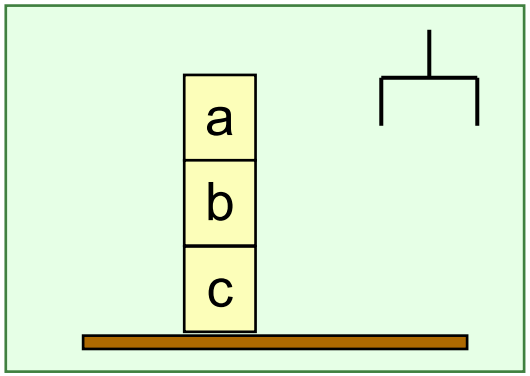
problem

```
(define (problem sussman)
   (:domain blocksworld)
   (:objects a b c)

   (:init  (ontable a)(ontable b)
           (clear c)(clear b)
           (on c a)(arm-empty)
   )
```



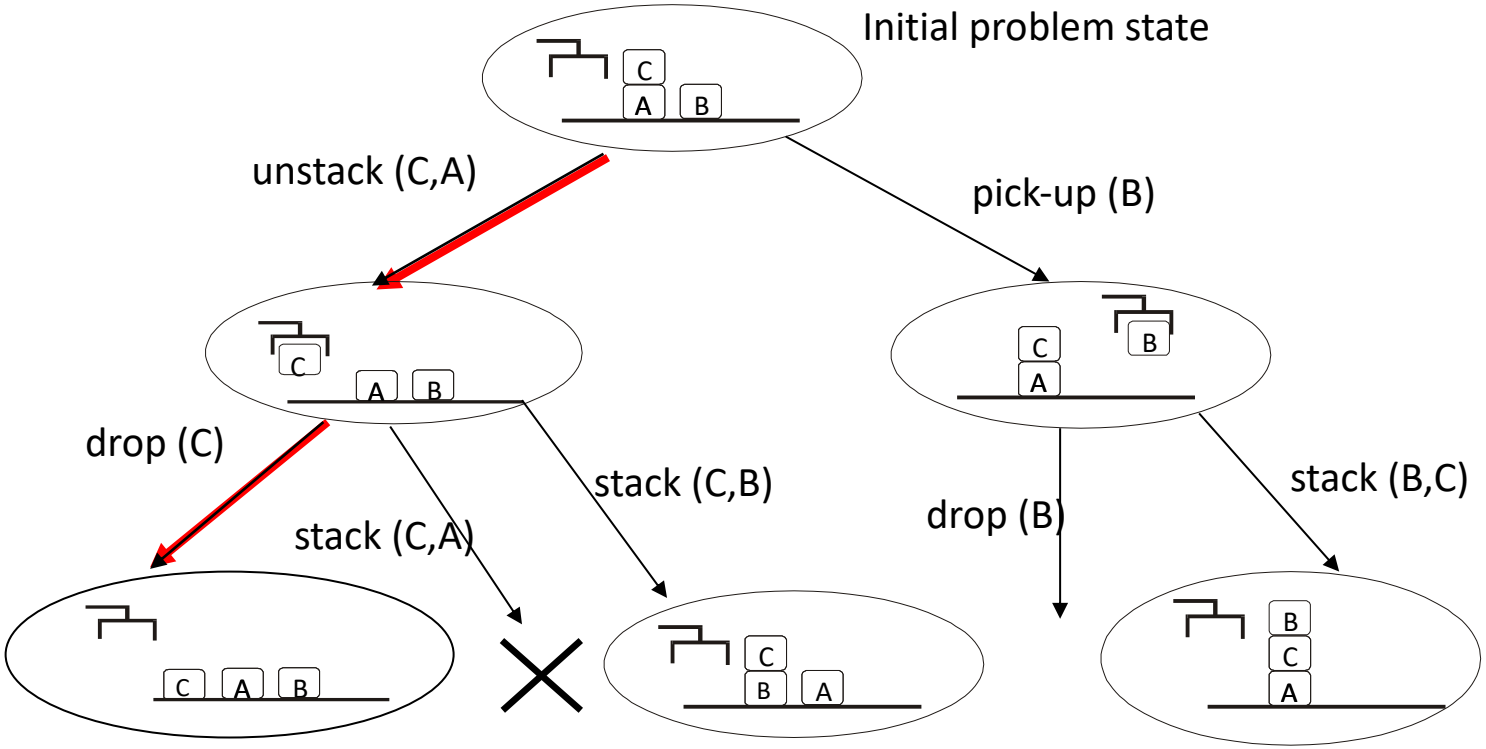

```
   (:goal  (on a b)(on b c)
   )
```



Reasoning to solve a control problem: solving process

Planning technique: heuristic planning

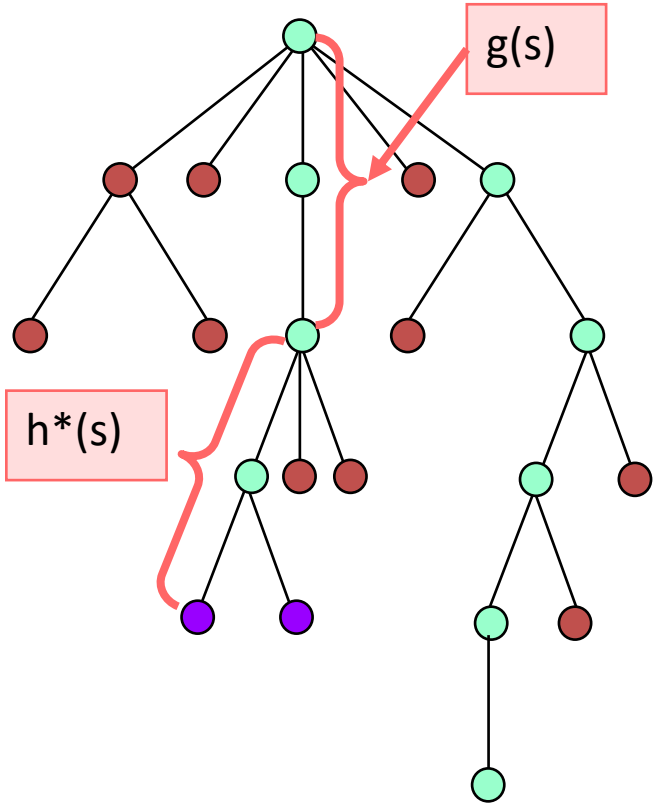
- state-based representation, forward search
- node-selection heuristic



Reasoning to solve a control problem: solving process

A* search

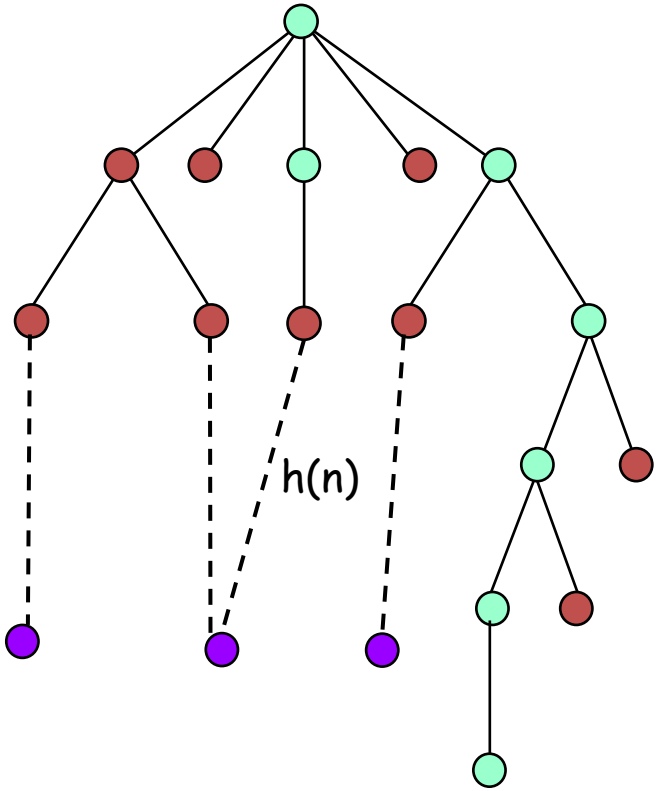
- For every state s, let
  - $g(s)$ = cost of the path from s_0 to s
  - $h^*(s)$ = least cost of all paths from s to goal nodes
  - $f^*(s) = g(s) + h^*(s)$ = least cost of all paths from s_0 to goal nodes that go through s
- Suppose $h(s)$ is an estimate of $h^*(s)$
  - Let $f(s) = g(s) + h(s)$
    - $f(s)$ is an estimate of $f^*(s)$
  - h is *admissible* if for every state s, $0 \leq h(s) \leq h^*(s)$



Reasoning to solve a control problem: solving process

A* search

- For every state s, let
  - $g(s)$ = cost of the path from s_0 to s
  - $h^*(s)$ = least cost of all paths from s to goal nodes
  - $f^*(s) = g(s) + h^*(s)$ = least cost of all paths from s_0 to goal nodes that go through s
- Suppose $h(s)$ is an estimate of $h^*(s)$
  - Let $f(s) = g(s) + h(s)$
    - $f(s)$ is an estimate of $f^*(s)$
  - h is *admissible* if for every state s, $0 \leq h(s) \leq h^*(s)$

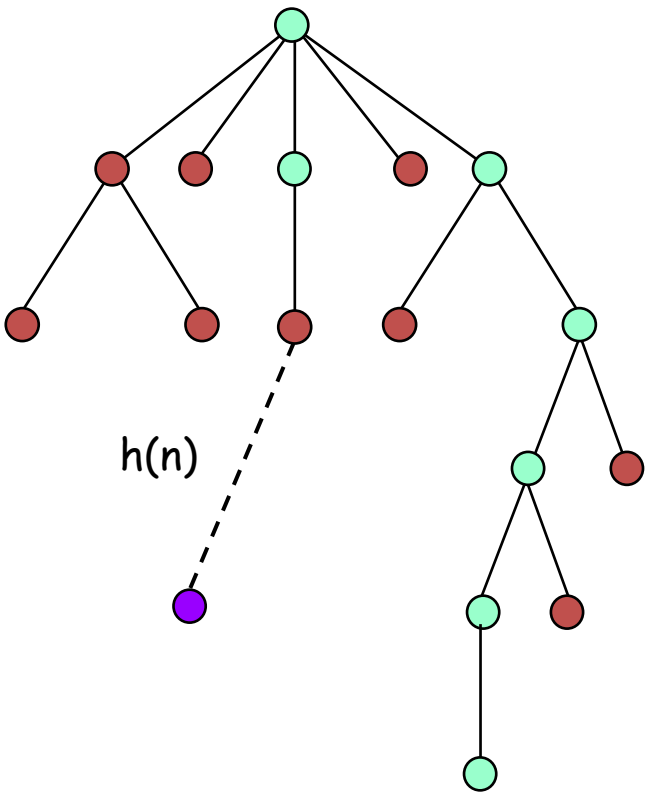


Reasoning to solve a control problem: solving process

Heuristics computed on a *relaxed planning graph*

- multi-layered graph calculated from a state and the domain actions
- ignore delete effects of actions
- a proposition level of the graph includes all the propositions that are potentially attainable at that level

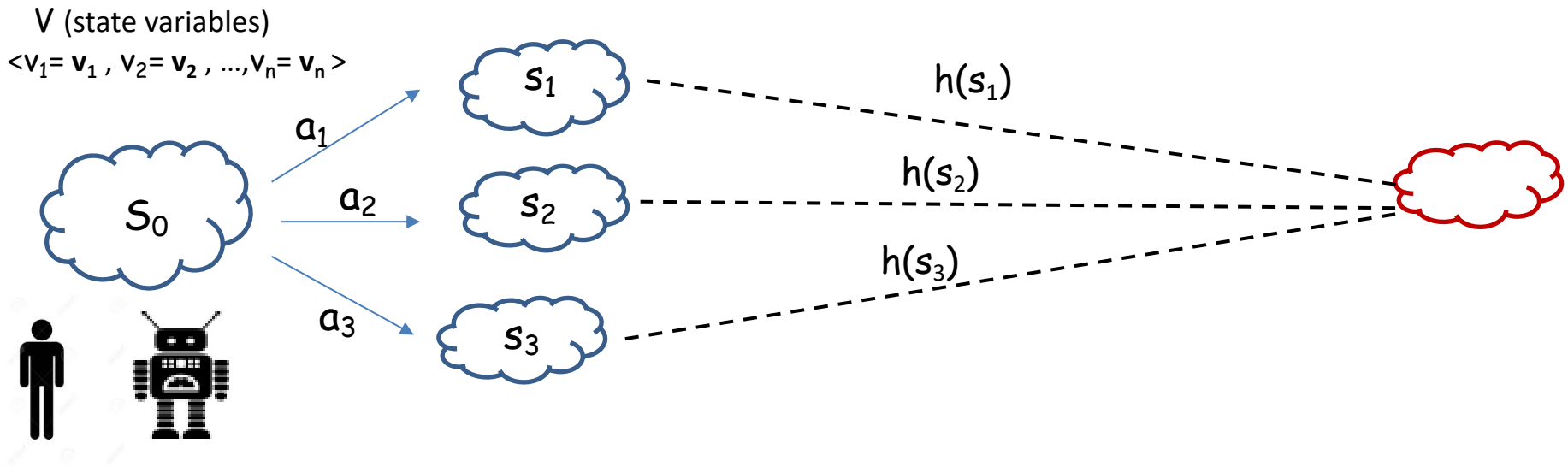
```
(:action stack
  :parameters (?ob - block ?underob - block)
  :precondition
      (and
          (clear ?underob)
          (holding ?ob))
  :effect
      (and
          (clear ?ob)
        (arm-empty)
        (on ?ob ?underob)
        (not (holding ?ob))
        (not (clear ?underob))
      )
)
```



Reasoning to solve a control problem: solving process

	P[0]	A[1]	P[1]	A[2]	P[2]	P[3]
1	on (C,A)	unstack(C,A)				
2	ontable(A)	pickup(B)				
3	ontable(B)			drop(C)		
4	arm-empty			stack(C,A)		
5	clear (C)			stack(C,B)		
6	clear (B)			drop(B)		
			holding (C)	stack(B,A)		
			holding (B)	stack(B,C)		
			clear (A)	pickup(A)		
					ontable(C)	
					on (C,B)	
					on (B,A)	
					on (B,C)	
					holding (A)	
						on (A,B)
						on (A,C)

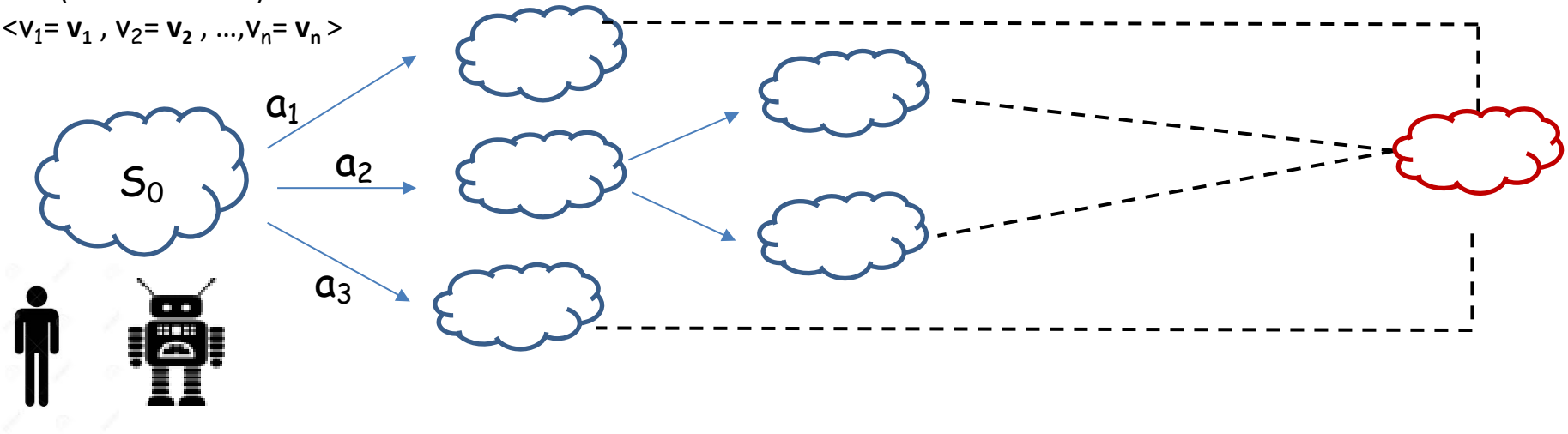
Decision-making control problema: solving process



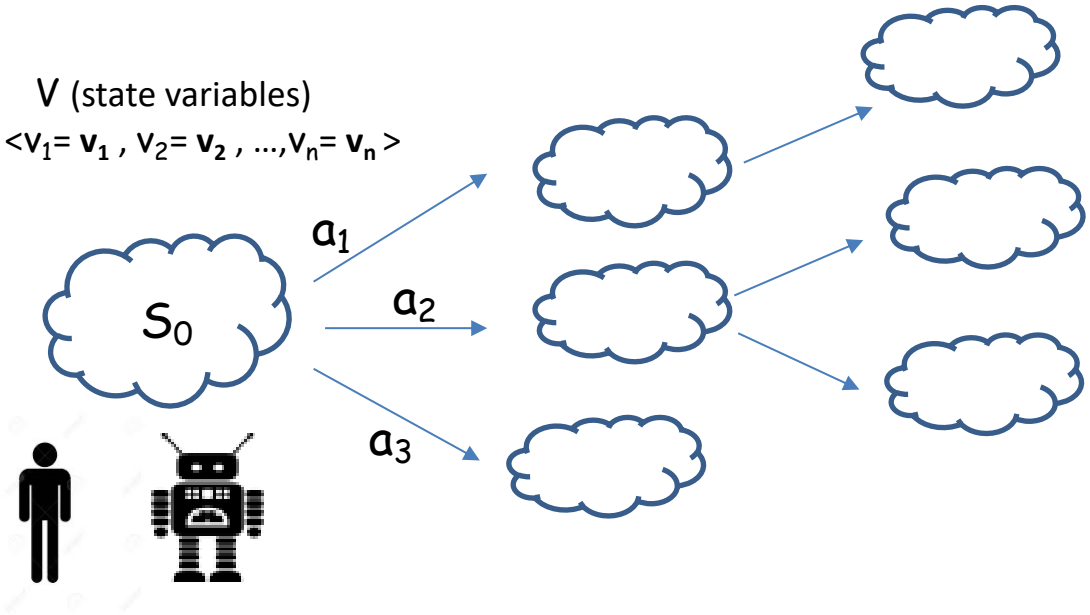
Decision-making control problem: solving process

V (state variables)

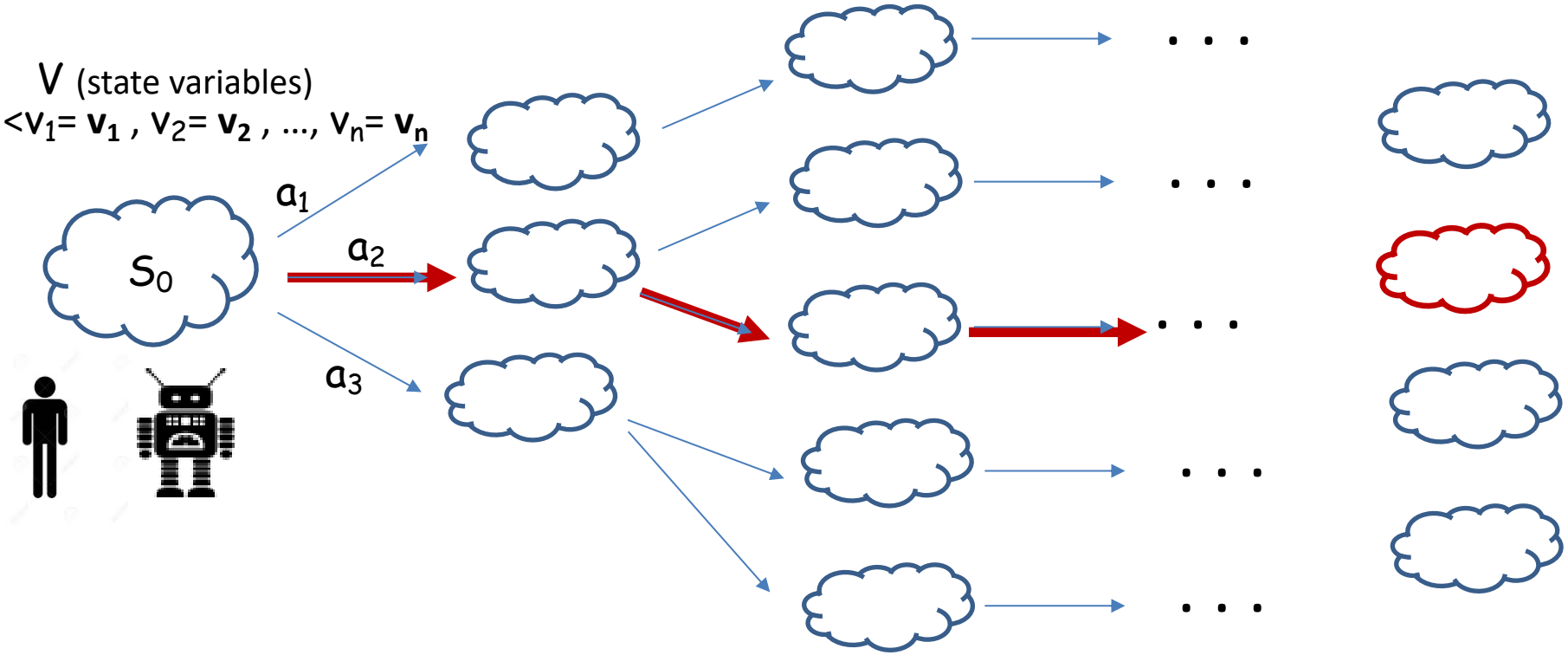
$<v_1 = \mathbf{v_1}, v_2 = \mathbf{v_2}, \ldots, v_n = \mathbf{v_n}>$



Decision-making control problem: solving process



Decision-making control problem: solving process



Reasoning to solve a control problem: wrap-up

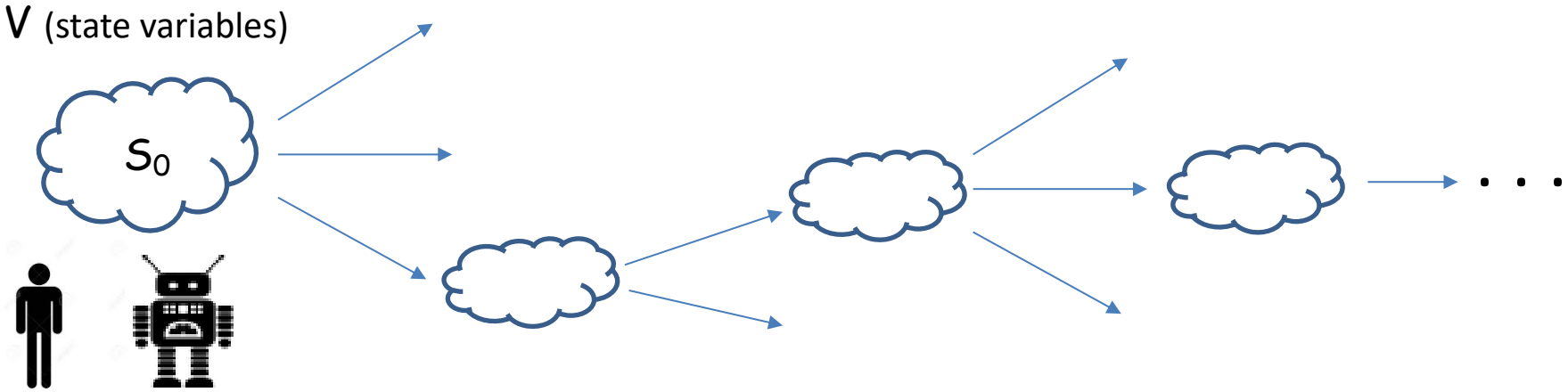
Characteristics:

- Solver is flexible, domain-independent
- Reasoning about states and actions (model of the world)
- Interpretability (as opposed to black-box models)
- Satisfaction task

- Requires a model specification (domain expert)
- Solution particular to a problem instance (lack of generalization)
- Computationally intractable (PSPACE-complete if we use a propositional language)

Learning to solve a control problem

- Learning = decision + apply (no simulation of the problem evolution)

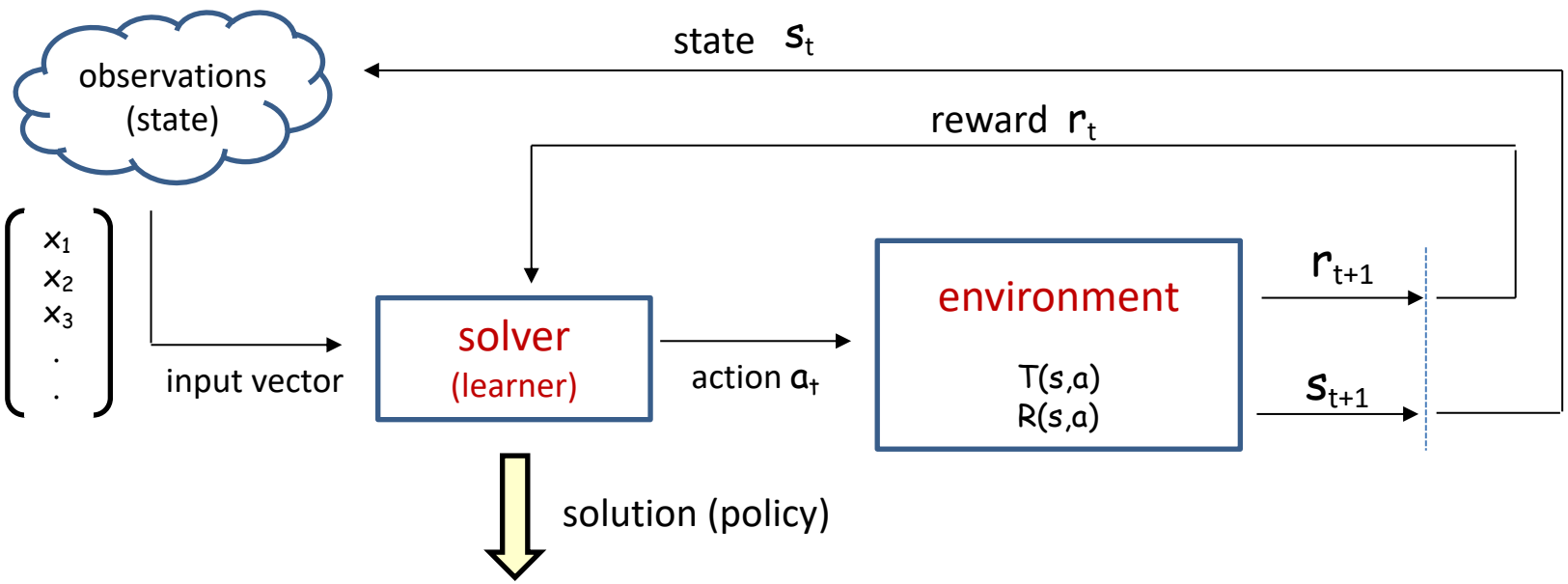


Learning to solve a control problem

- Learning = decision + apply (no simulation of the problem evolution)
- A model specification is not always available (model-free approach)

- Learning from interaction with the environment (learning to plan/act)
- A solver (learner) learns control from experience

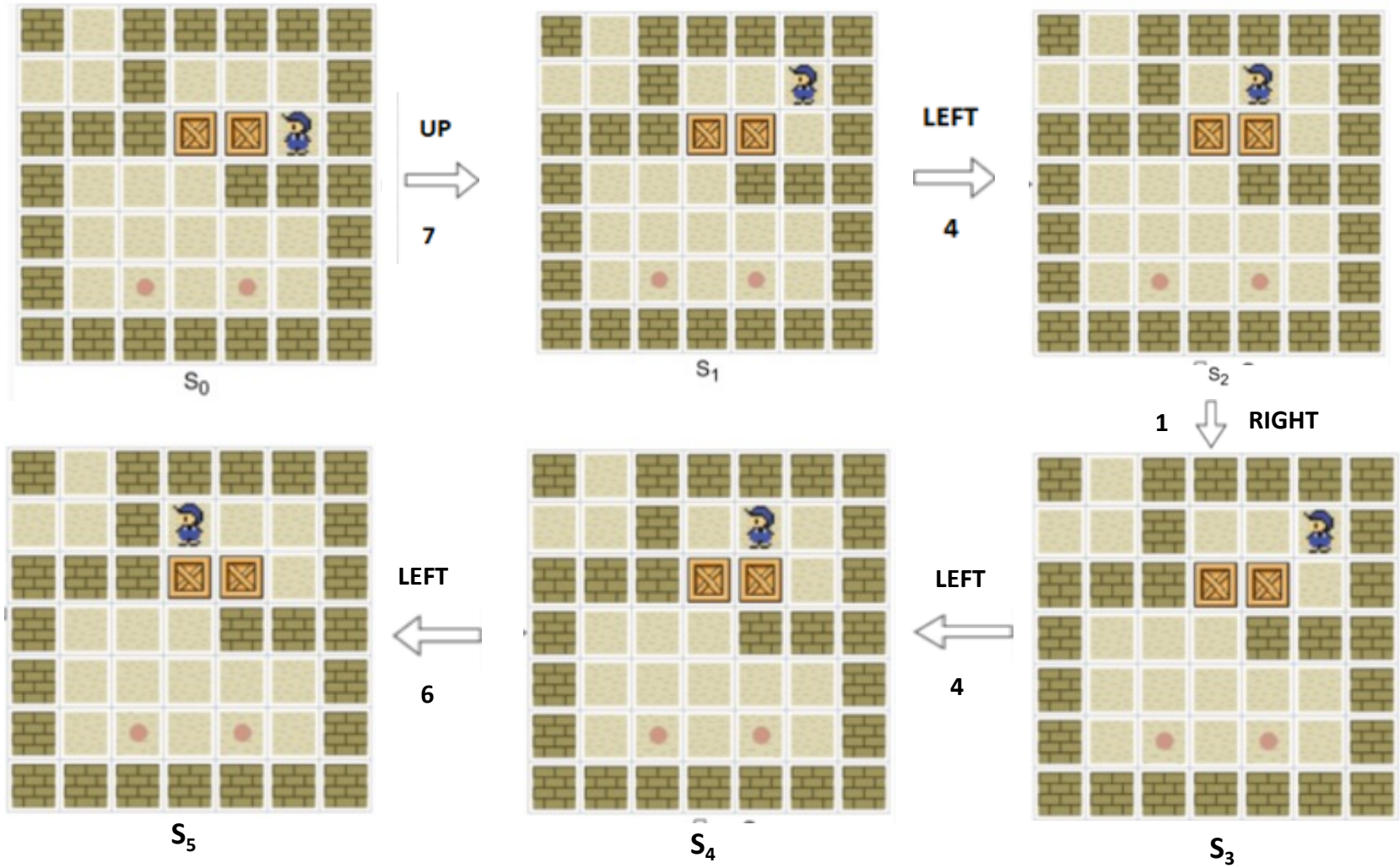
REINFORCEMENT LEARNING



$\pi: s \rightarrow a$

$S \times A$

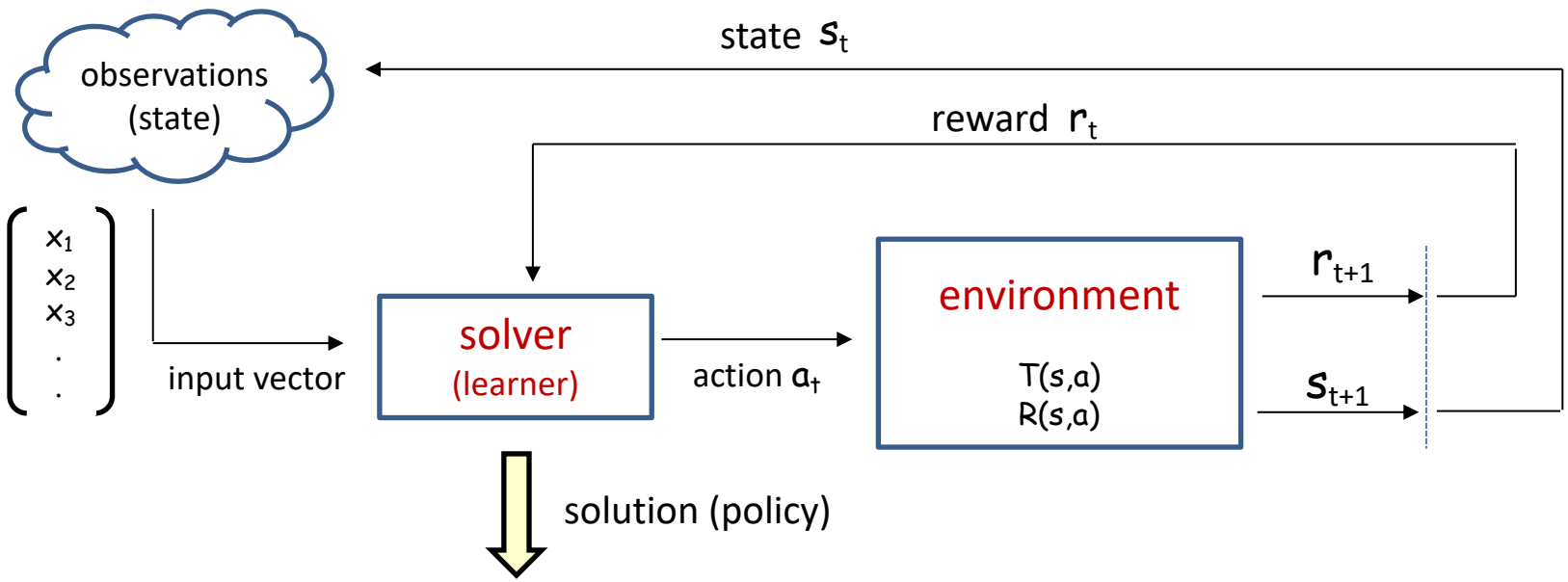
Learning to solve a control problem



Learning to solve a control problem: environment

Stochastic environments:

- uncertainty in the result of the agent actions
- most popular models to deal with uncertainty are based on the Markov assumption
- Markov Decision Processes (MDP)



Learning to solve a control problem: environment

Markov property:

- the rewards received from the environment and also the state transition achieved by the environment at time **t+1** depend probabilistically on only the state/action pair at time **t**
- the future state and reward can be exclusively defined on the current state and the action applied in such state

$$Pr(s_{t+1}=s', r_{t+1}=r \mid s_t, a_t)$$

Markov Decision Process (MDP): $P = \langle S, A, T, R \rangle$

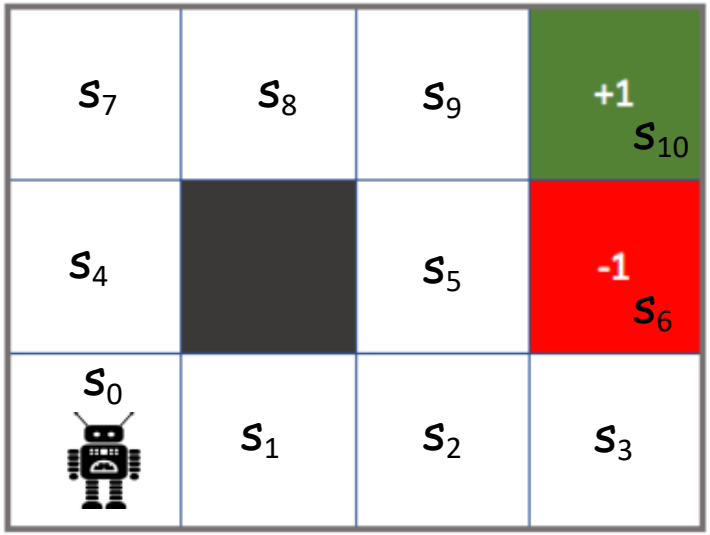
- S: set of states
- A: set of actions
- T: $S \times A \rightarrow Pr(S)$:
  - probability distribution over the set S
  - $p(s,a,s')$: probability of reaching state s' when taking action a in state s
- R: $S \times A \rightarrow \Re$: reward received when taking action a in state s

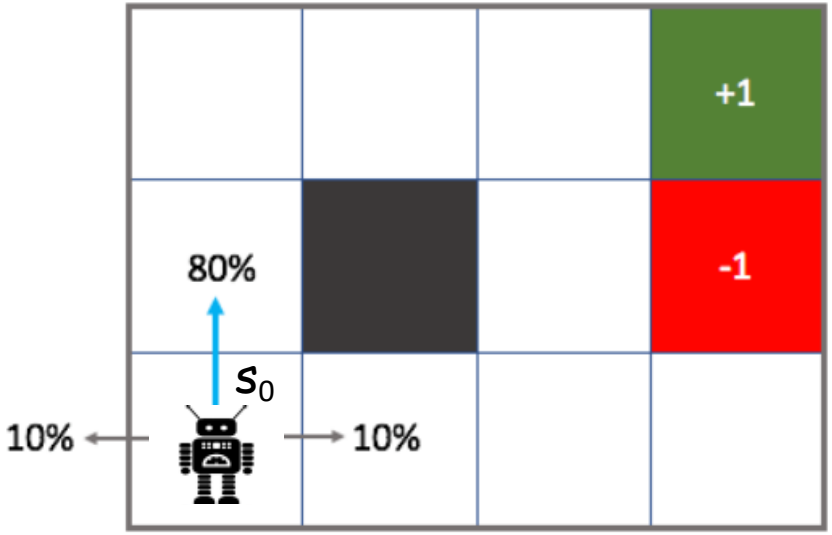
Policy: $\Pi(a|s)$

- The probability of taking action a in state s
- A function that determines the behaviour of the agent

Learning to solve a control problem: environment

https://www.jeremyjordan.me/markov-decision-process/



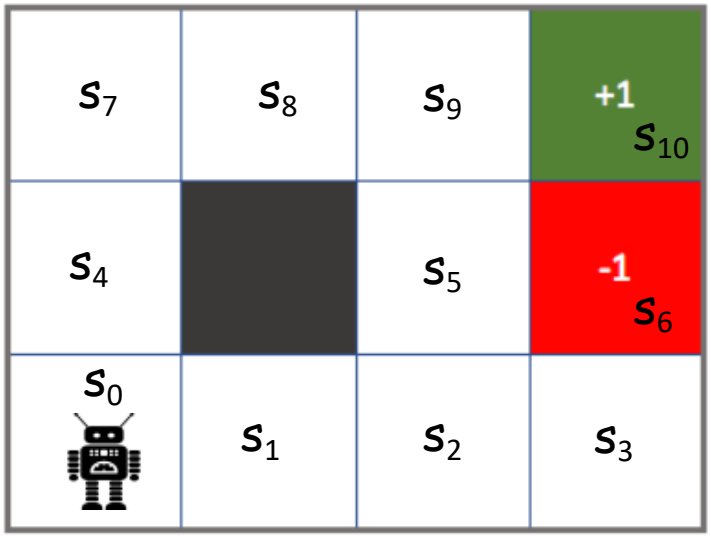


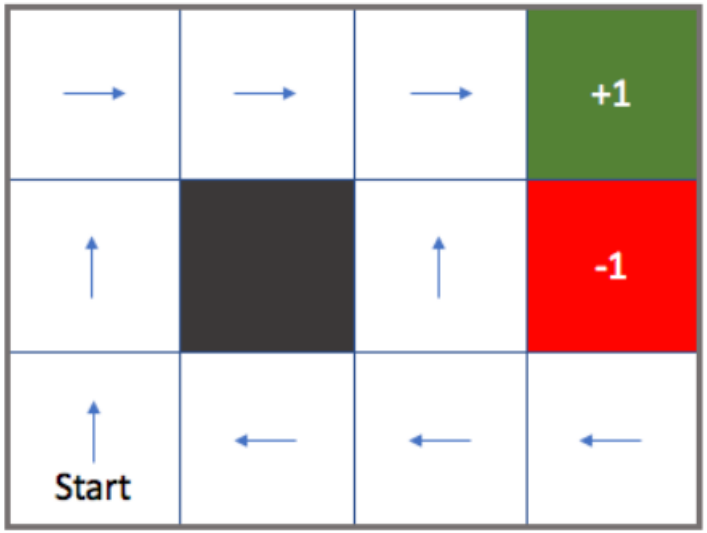
- S: every square in the grid except the black square:$\{s_1, s_2, \ldots, s_{10}\}$
- A: {up, down, right, left}
- $R(s_9, \text{right})=+1$ $R(s_0,\text{up})=0.04$
  $R(s_3,\text{up})= -1$ …

$p(s_0,\text{up},s_4)=0.8$
$p(s_0,\text{up},s_1)=0.1$
$p(s_0,\text{up},s_0)=0.1$

Learning to solve a control problem: environment

https://www.jeremyjordan.me/markov-decision-process/





R(state) = -0.04

Policy examples

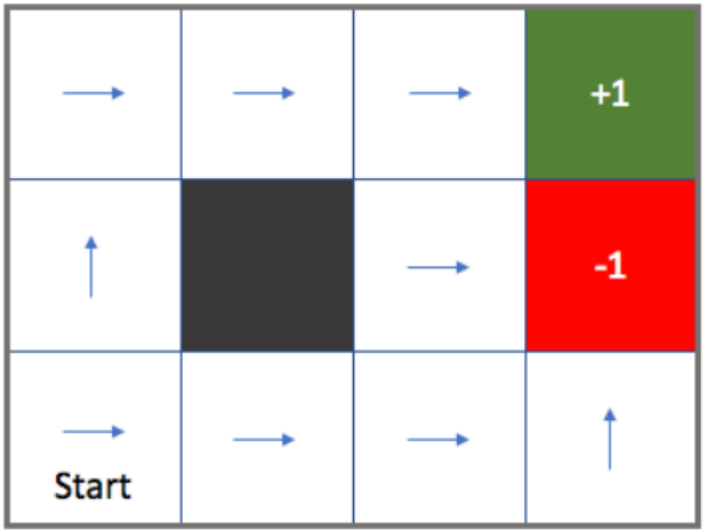
$\Pi(up|s_0)=0.95$
$\Pi(right|s_0)=0.05$

$\Pi(up|s_2)=0.75$
$\Pi(right|s_2)=0.1$
$\Pi(left|s_2)=0.15$

Optimal policy where each white state has a reward of -0.04

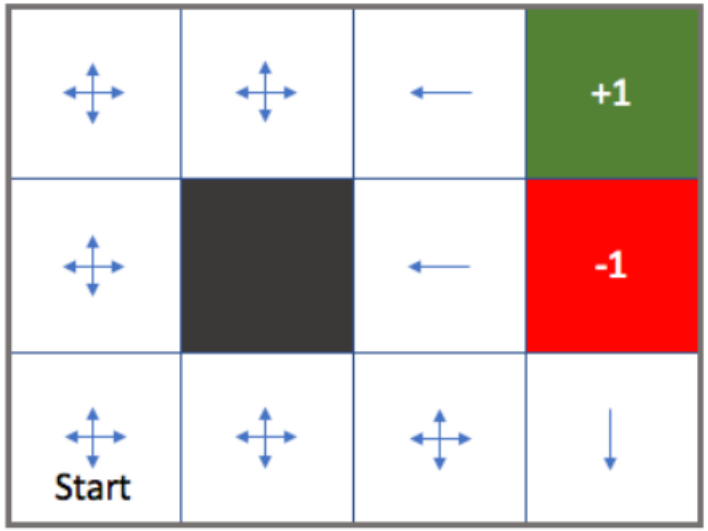
Learning to solve a control problem: environment

https://www.jeremyjordan.me/markov-decision-process/



R(state) = -2

Optimal policy where each white state has a reward of -2



R(state) = +2

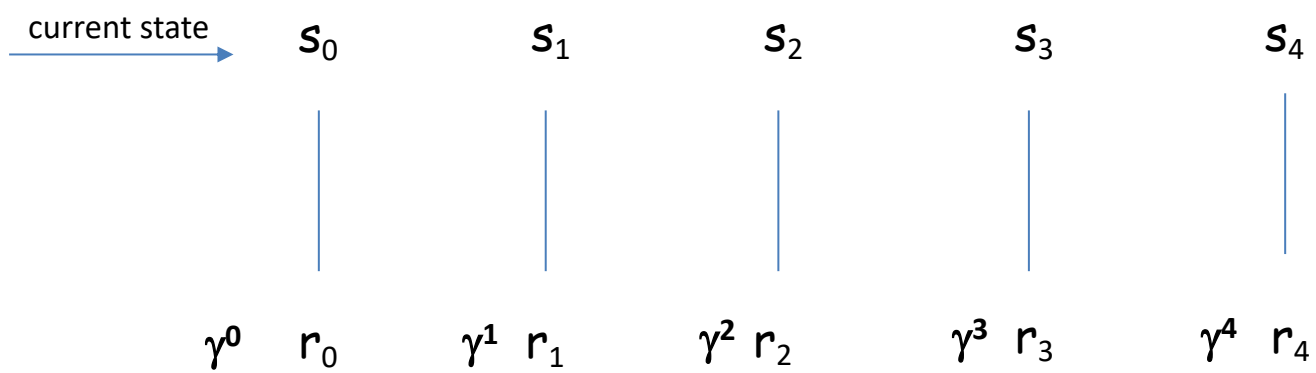
Optimal policy where each white state has a reward of +2

Learning to solve a control problem: solver

It would be helpful to calculate the usefulness (value) of a sequence of states such that our 'view of the future' is not limited to the immediate reward of the next direct action.

Idea: explore what is ahead so that we can move toward the direction of positive reward and move away from negative reward

Utility of a sequence of states:



γ discount factor
favor more immediate rewards – deemphasize the rewards that are more distant
We cannot be sure future states are reached due to the stochastic nature of the transition

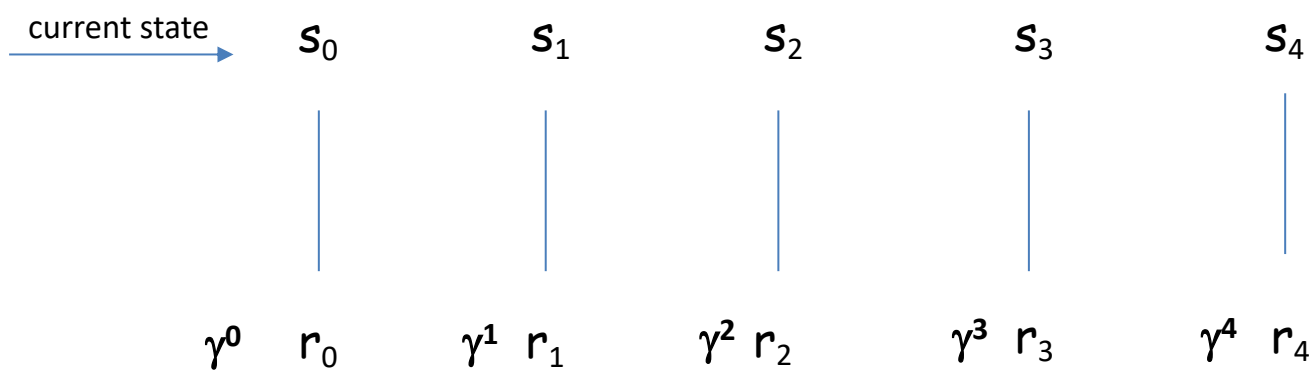
Learning to solve a control problem: solver

utility(s_0,s_1,…,s_n)= $\sum_{t=0}^{\infty} \gamma^t r_t$

$$V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\right]$$

expected reward-value of state s
state-value of a state s

Utility of a sequence of states:



γ discount factor
favor more immediate rewards – deemphasize the rewards that are more distant
We cannot be sure future states are reached due to the stochastic nature of the transition

Learning to solve a control problem: solver

$$V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\right]$$

expected reward-value of state ***s***
state-value of a state ***s***

$$\boldsymbol{V^{\pi}}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\right]$$

expected reward-value of state ***s*** for policy π

$$\boldsymbol{\pi}^* = argmax_{\pi}\, E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi\right]$$

optimal policy

$$\boldsymbol{V}^*(s) = max_{\pi}\ V^{\pi}(s)$$

optimal state-value function

Learning to solve a control problem: solver

$$\boldsymbol{V^{\pi}}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\right]$$

expected state-value of state **s** for policy π

$$\boldsymbol{Q^{\pi}}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s, a_t = a\right]$$

expected value after taking action **a** in state **s** for policy π

action-value of a pair state-action

$$\boldsymbol{Q}^*(s,a) = max_{\pi}\ Q^{\pi}(s,a)$$

$$\boldsymbol{V}^*(s) = max_{a \in A}\ Q^*(s,a)$$

$$\boldsymbol{Q}^*(s,a) = \sum_{s'} p(s,a,s')\,[R(s,a) + \gamma V^*(s')]$$

$$\boldsymbol{Q}^*(s,a) = \sum_{s'} p(s,a,s')\,[R(s,a) + \gamma max_{a'} Q^*(s',a')]$$

Bellman equations

Learning to solve a control problem: environment

So far, we have complete knowledge over the system dynamics:

- reward of each state $R(s,a)$
- a probabilistic model for how the agent changes its state $T(s,a)$

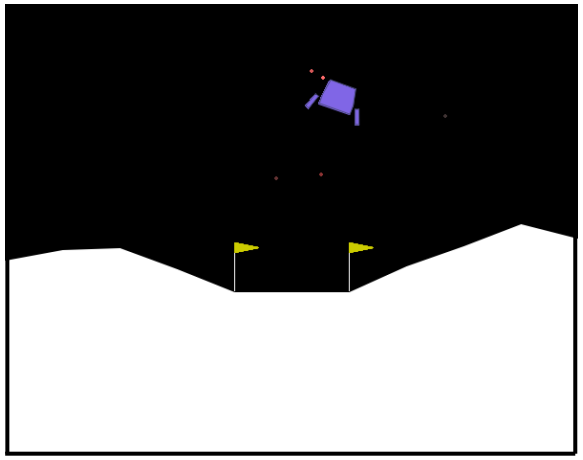
- T and R are known
- The agent has access to the MDP model
- Known environment = (stochastic) planning

What can we do if T and R are unknown? That is, the agent does not have access to the underlying systems dynamics or MDP model

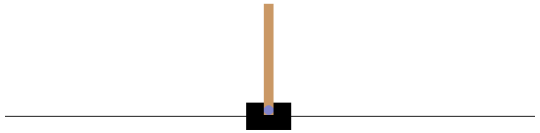
Direct Reinforcement Learning

Learning to solve a control problem: environment

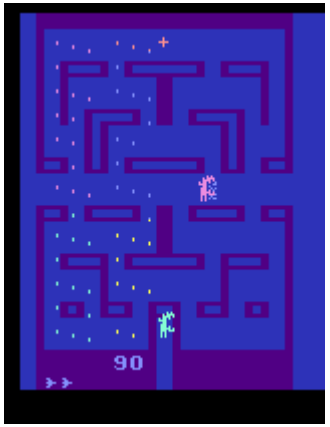
Open AI Gymnasium (https://gymnasium.farama.org/)
API for RL and collection of environments



Luna Lander



Cart Pole

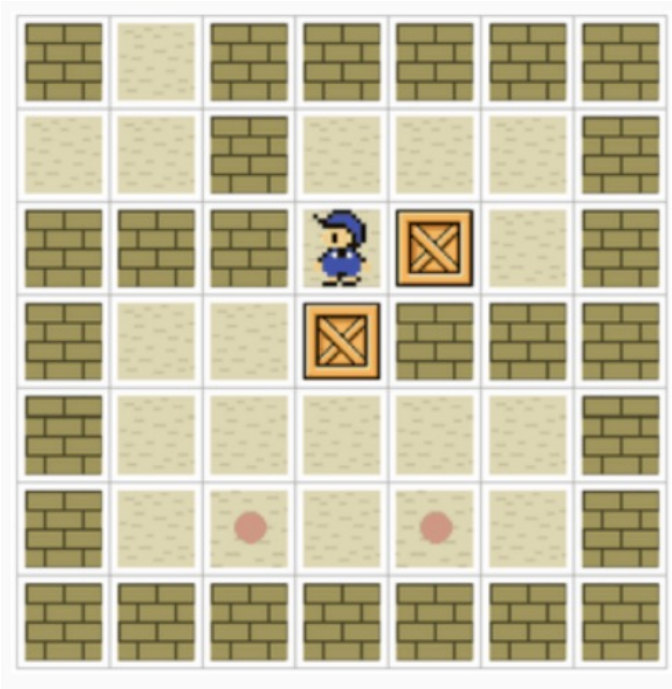


Atari games (Alien)

Frozen Lake

Learning to solve a control problem: environment

PDDLGym (https://github.com/tomsilver/pddlgym)



```
import pddlgym
import imageio

env = pddlgym.make("PDDLEnvSokoban-v0")
obs, debug_info = env.reset()
img = env.render()
imageio.imsave("frame1.png", img)
action = env.action_space.sample(obs)
obs, reward, done, debug_info = env.step(action)
img = env.render()
imageio.imsave("frame2.png", img)
```

Learning to solve a control problem: solver

Free-model methods:

- update the value of pairs (s_t,a_t) from trajectories $\langle s_0,a_0,r_1,s_1,a_1,r_2, \ldots, s_T\rangle$
- Monte-Carlo: update the value considering the sum of reward of the complete trajectory (episode)
- Temporal Difference (TD): update the value considering individual transitions (steps inside the episode)

Q-learning algorithm:

- standard free-model method
- a specific TD-algorithm to learn the Q function
- it is a control method to estimate $Q^*(s,a)$

$$\widehat{\pi}(s) = \arg\max_a \widehat{Q}(s, a)$$

Learning to solve a control problem: Q-learning algorithm

- it learns iteratively the optimal Q value function using the Bellman optimality equations
- we store all the Q-values in a table that we will update at each time step using the Q-learning iteration

$$S \times A$$
$$\begin{bmatrix} \quad \end{bmatrix}$$

$$\boldsymbol{Q}^*(s,a) = \sum_{s'} p(s,a,s') \left[R(s,a) + \gamma max_{a'} Q^*(s',a')\right]$$

$$\widehat{Q}(s_t, a_t) \leftarrow (1-\alpha_t)\widehat{Q}(s_t, a_t) + \alpha_t \left[r_{t+1} + \gamma \max_a \widehat{Q}(s_{t+1}, a)\right]$$

α_t : learning rate

```
Initialize Q(s,a), ∀s ∈ S, a ∈ A(s), arbitrarily, and Q(terminal-state, ·) = 0
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        Q(S,A) ← Q(S,A) + α[R + γ max_a Q(S',a) − Q(S,A)]
        S ← S';
    until S is terminal
```

For Q-learning to converge, it is necessary to visit every state or every state-action a large number of times

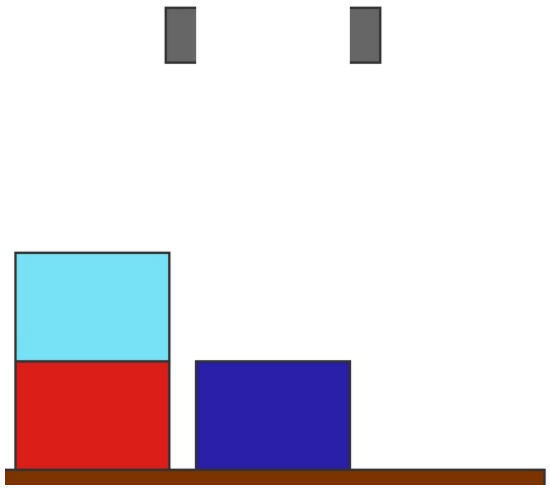
Learning to solve a control problem: Q-learning algorithm

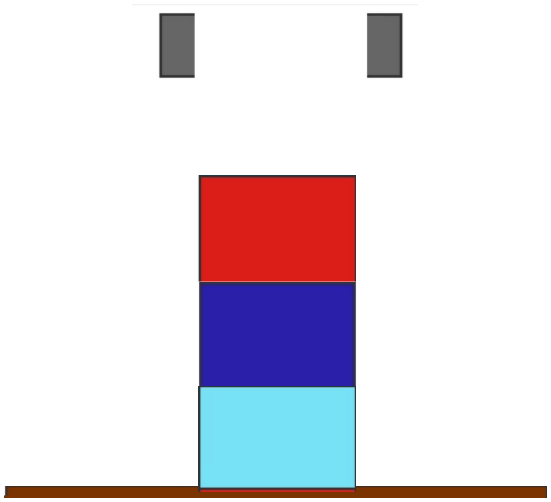
```
Initialize Q(s,a), ∀s ∈ S, a ∈ A(s), arbitrarily, and Q(terminal-state, ·) = 0
Repeat (for each episode):
   Initialize S
   Repeat (for each step of episode):
      Choose A from S using policy derived from Q (e.g., ε-greedy)
      Take action A, observe R, S'
      Q(S,A) ← Q(S,A) + α[R + γ max_a Q(S',a) − Q(S,A)]
      S ← S';
   until S is terminal
```

Exploitation vs exploration:

- Should I keep doing what I do or should I try something else?
- At the beginning the agent has no idea about the environment, so it is more likely to explore new things than to exploit their knowledge
- The agent will get more and more information about how the environment works and then it is more likely to exploit

Learning to solve a control problem: Q-learning algorithm

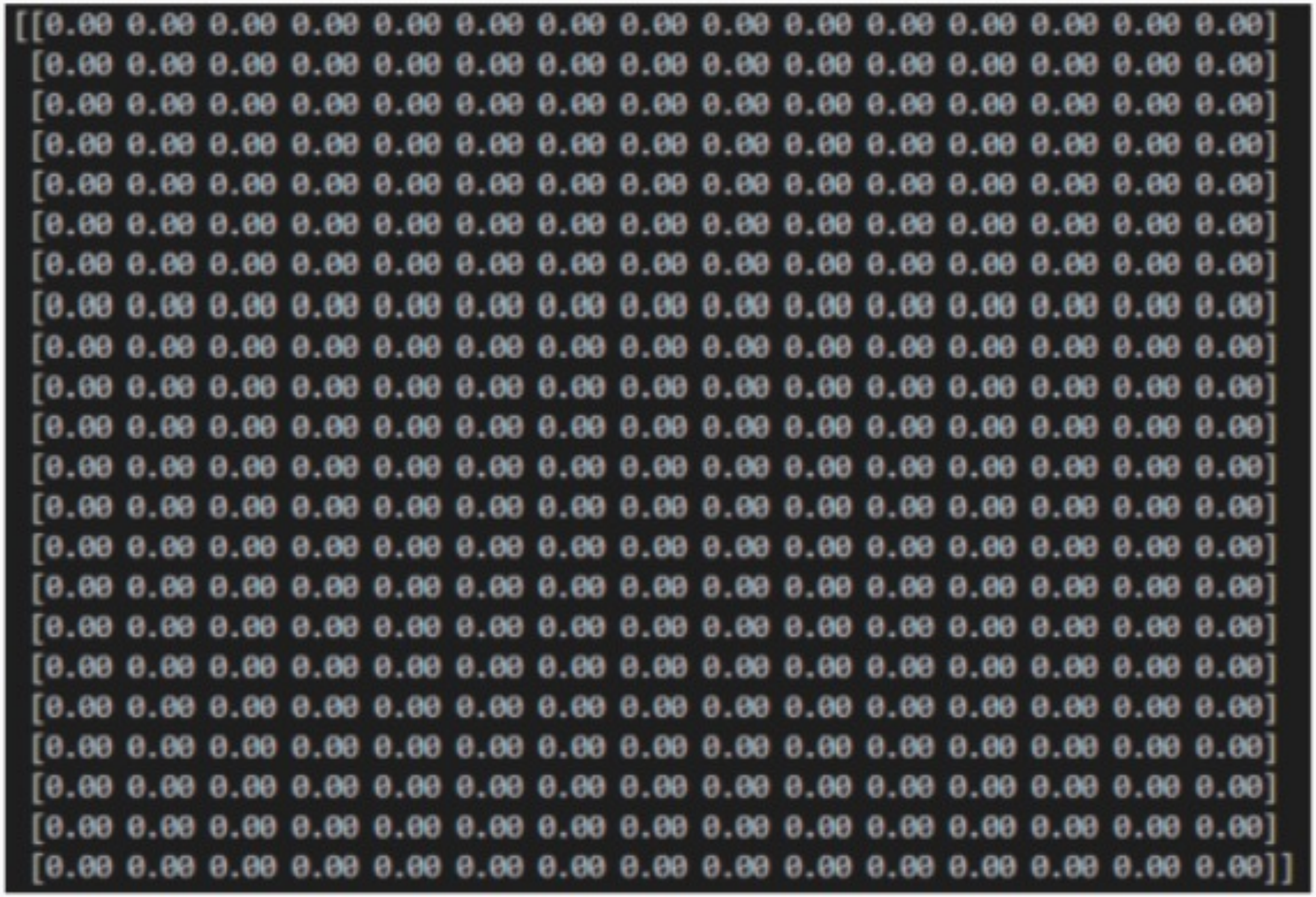




18 actions:

```
(pickup R)(pick-up LB)(pickup DB)
(stack R LB)(stack LB R)(stack R DB) ….
(unstack R LB)(unstack LB R) …
(drop R) …
```

Learning to solve a control problem: Q-learning algorithm



```
[[0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]]
```

Learning to solve a control problem: Q-learning algorithm

```
[[0.72 0.69 0.57 0.79 0.71 0.70 0.66 0.69 0.71 0.65 0.61 0.69 0.73 0.71 0.72]
 [0.51 0.54 0.60 0.54 0.54 0.37 0.50 0.54 0.68 0.55 0.47 0.57 0.49 0.54 0.53]
 [0.27 0.33 0.23 0.35 0.31 0.12 0.29 0.52 0.25 0.32 0.14 0.18 0.18 0.26 0.22]
 [0.11 0.12 0.13 0.13 0.06 0.10 0.07 0.14 0.15 0.13 0.09 0.07 0.11 0.09 0.20]
 [0.08 0.13 0.05 0.08 0.07 0.09 0.08 0.10 0.10 0.10 0.08 0.05 0.09 0.10 0.08]
 [0.62 0.58 0.55 0.53 0.45 0.58 0.53 0.55 0.57 0.49 0.60 0.32 0.47 0.47 0.71]
 [0.26 0.51 0.10 0.19 0.21 0.17 0.15 0.20 0.13 0.29 0.26 0.27 0.24 0.22 0.25]
 [0.13 0.09 0.11 0.10 0.13 0.15 0.09 0.09 0.24 0.13 0.11 0.16 0.09 0.11 0.11]
 [0.06 0.05 0.09 0.05 0.06 0.09 0.06 0.15 0.10 0.05 0.06 0.08 0.08 0.04 0.08]
 [0.39 0.61 0.48 0.13 0.41 0.46 0.26 0.52 0.48 0.52 0.48 0.39 0.29 0.48 0.42]
 [0.60 0.76 0.74 0.82 0.71 0.72 0.90 0.73 0.74 0.78 0.78 0.64 0.67 0.46 0.72]
 [0.86 0.81 0.95 0.87 0.74 0.85 0.86 0.85 0.86 0.75 0.86 0.84 0.81 0.90 0.88]
 [0.73 0.72 0.74 0.85 0.72 0.99 0.79 0.67 0.61 0.66 0.50 0.78 0.67 0.92 0.63]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.48 0.52 0.47 0.37 0.52 0.51 0.49 0.57 0.47 0.44 0.50 0.50 0.53 0.49 0.49]
 [0.37 0.41 0.39 0.43 0.43 0.43 0.35 0.34 0.30 0.40 0.44 0.42 0.32 0.50 0.43]
 [0.30 0.20 0.30 0.27 0.30 0.30 0.27 0.24 0.37 0.45 0.33 0.24 0.29 0.25 0.31]
 [0.40 0.29 0.54 0.48 0.46 0.43 0.38 0.42 0.55 0.78 0.10 0.25 0.35 0.34 0.31]
 [0.08 0.07 0.12 0.13 0.10 0.11 0.05 0.08 0.13 0.06 0.08 0.12 0.08 0.34 0.07]
 [0.07 0.07 0.06 0.02 0.11 0.04 0.07 0.11 0.10 0.11 0.08 0.02 0.01 0.10 0.28]
 [0.04 0.05 0.04 0.06 0.06 0.04 0.05 0.05 0.04 0.10 0.03 0.05 0.05 0.02 0.04]
 [0.00 0.10 0.00 0.01 0.00 0.00 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.00 0.02]]
```

Learning to solve a control problem: Q-learning algorithm

```
[[0.96 0.96 0.95 0.97 0.96 0.96 0.96 0.96 0.96 0.96 0.95 0.96 0.96 0.96 0.96]
 [0.95 0.95 0.95 0.95 0.95 0.94 0.95 0.95 0.96 0.95 0.95 0.95 0.94 0.95 0.95]
 [0.94 0.93 0.93 0.93 0.93 0.93 0.93 0.95 0.93 0.93 0.81 0.92 0.93 0.93 0.93]
 [0.50 0.46 0.60 0.64 0.42 0.70 0.80 0.58 0.64 0.67 0.68 0.59 0.70 0.77 0.91]
 [0.25 0.71 0.07 0.10 0.31 0.35 0.35 0.20 0.34 0.29 0.26 0.29 0.23 0.33 0.18]
 [0.95 0.95 0.95 0.95 0.94 0.95 0.95 0.95 0.95 0.95 0.95 0.94 0.95 0.95 0.96]
 [0.94 0.95 0.92 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94]
 [0.87 0.90 0.82 0.85 0.79 0.77 0.79 0.87 0.94 0.82 0.80 0.88 0.69 0.86 0.87]
 [0.55 0.53 0.52 0.40 0.53 0.48 0.49 0.84 0.68 0.49 0.40 0.36 0.59 0.32 0.49]
 [0.94 0.95 0.94 0.93 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94 0.94]
 [0.96 0.97 0.97 0.97 0.97 0.97 0.98 0.97 0.97 0.97 0.97 0.97 0.97 0.96 0.97]
 [0.98 0.98 0.99 0.98 0.98 0.98 0.98 0.98 0.98 0.97 0.98 0.98 0.98 0.98 0.98]
 [0.99 0.99 0.99 0.99 0.99 1.00 0.99 0.99 0.98 0.99 0.99 0.99 0.99 0.99 0.99]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.93 0.94 0.93 0.92 0.94 0.94 0.91 0.95 0.94 0.94 0.93 0.92 0.94 0.93 0.93]
 [0.79 0.90 0.87 0.83 0.85 0.87 0.88 0.86 0.85 0.89 0.89 0.88 0.85 0.94 0.90]
 [0.68 0.61 0.64 0.72 0.77 0.73 0.70 0.69 0.60 0.88 0.73 0.65 0.69 0.70 0.74]
 [0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.96 0.97 0.95 0.96 0.96 0.96 0.96]
 [0.82 0.90 0.89 0.88 0.89 0.85 0.85 0.84 0.89 0.80 0.83 0.82 0.87 0.94 0.85]
 [0.91 0.92 0.94 0.93 0.91 0.84 0.94 0.90 0.95 0.90 0.94 0.88 0.93 0.92 0.96]
 [0.75 0.64 0.74 0.79 0.82 0.68 0.71 0.79 0.56 0.91 0.75 0.71 0.74 0.61 0.51]
 [0.46 0.94 0.78 0.84 0.79 0.81 0.82 0.76 0.71 0.75 0.80 0.86 0.77 0.81 0.77]]
```

Learning to solve a control problem

Generalized planning with RL:

- RL can also be used to discover a strategy for a wide variety of problem within a domain
- In Generalized Planning, we want to achieve solutions that potentially solve any problem of a domain. Hence, we lose optimality.
- A well-known general strategy for the ***Blocksworld*** domain is to place all blocks on the table and stack them in the right order.

RL approximators:

- Design RL elements by means of a function approximator: NN, random forest, multivariate regression, … ➔ Deep Reinforcement Learning

Learning to solve a control problem: wrap-up

Characteristics:

- Uncertainty in the environment, unknown environments
- Generalized solutions
- Model-free approach (limited ‘view of the future’)
- Optimization task

- No interpretability
- The curse of dimensionality
- Sparse rewards (need of a large number of episodes to reach a solution)
- Solutions dependent on the training samples

References

Malik Ghallab, Dana Nau, and Paolo Traverso. ***Automated Planning. Theory and Practice*.** Morgan Kaufmann, (2004).

Avrim Blum and Merrick L. Furst. ***Fast Planning Through Planning Graph Analysis***. Artif. Intell., 90(1-2), 281–300, (1997)

J. Hoffmann, B. Nebel. ***The FF planning system: Fast plan generation through heuristic search***. J. Artif. Intell. Res. 14 (2001) 253–302.

Tom Silver and Rohan Chitnis. **PDDLGym: Gym Environments from PDDL Problems**. International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop, 2020.

Richard S. Sutton, Andrew G. Barto. ***Reinforcement learning - an Introduction***. Adaptive computation and machine learning, MIT Press 1998, ISBN 978-0-262-19398-6, pp. I-XVIII, 1-322

Sergio Jiménez Celorrio, Javier Segovia Aguas, and Anders Jonsson. ***A review of generalized planning***. Knowl. Eng. Rev., 34, e5, (2019)

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine***. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning***. in NIPS, pp. 15220–15231, (2019).

Asociación Española para la Inteligencia Artificial (**AEPIA**)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Planificación Automática y Aprendizaje por Refuerzo para problemas de toma de decisión

Escuela de Verano de Inteligencia Artificial

Eva Onaindía
Valencian Research Artificial Intelligence Institute (VRAIN)
Universitat Politècnica de València

14 Junio 2023