M.Sc. in Computer Science and Engineering

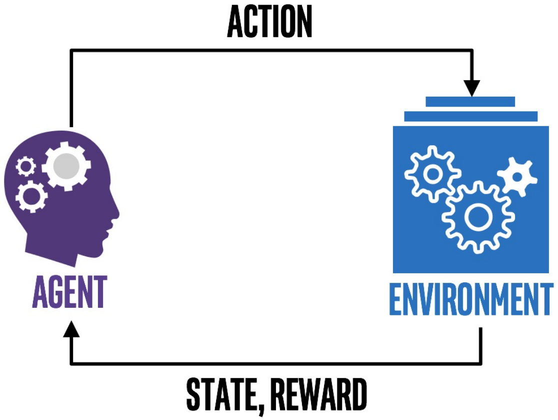
Non-Cooperative Configurable Markov Decision Processes

Alessandro Concetti

Supervisor: Prof. Marcello Restelli

Co-supervisors: Dott. Alberto Metelli, Dott.ssa Giorgia Ramponi

Reinforcement Learning


ACTION

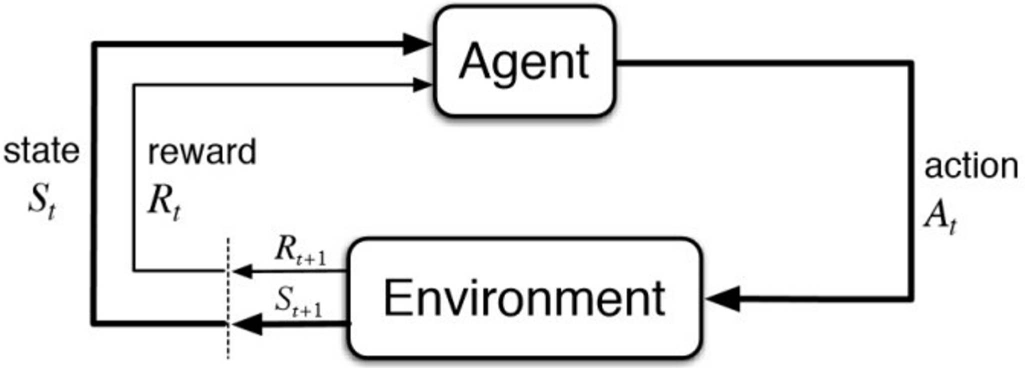
AGENT

ENVIRONMENT

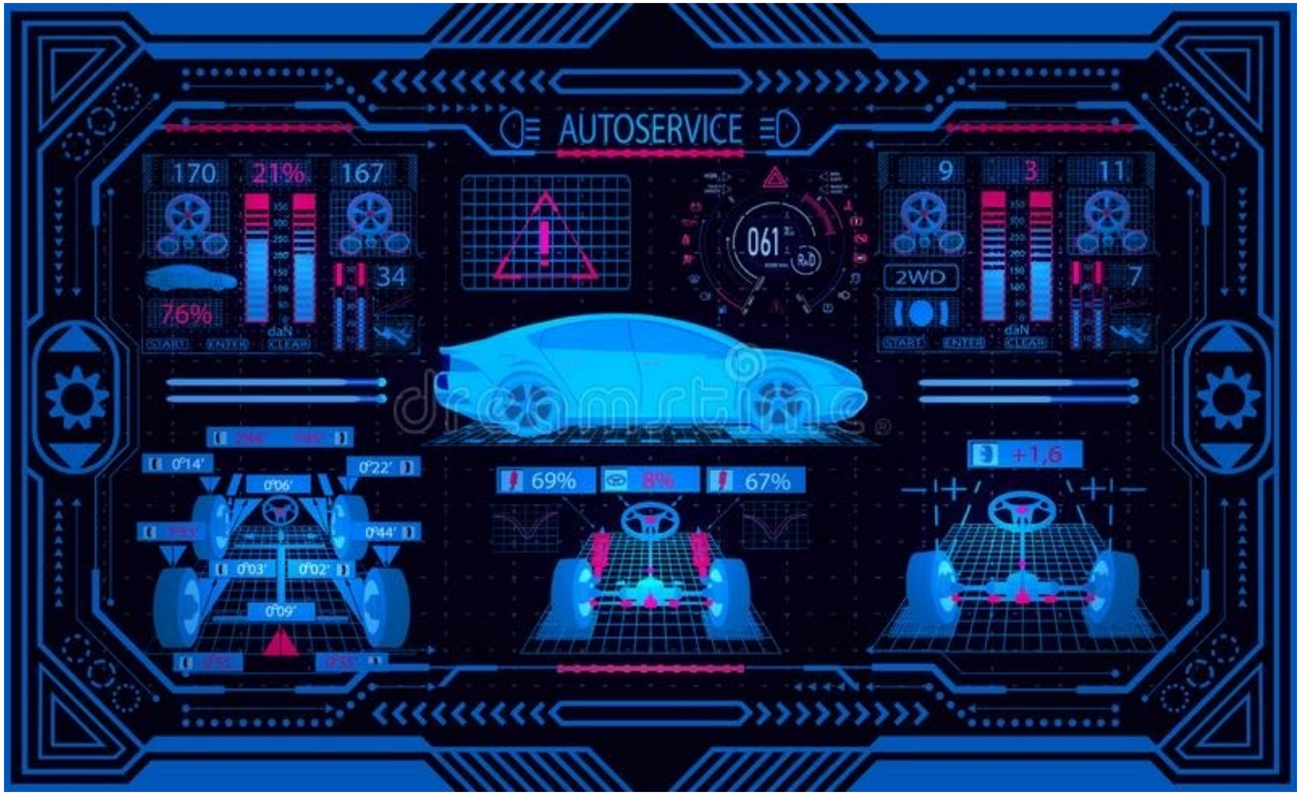
STATE, REWARD

Markov Decision Process

A **Markov Decision Process (MDP)** [Puterman, 2014] is a mathematical framework for modelling sequential decision making problems.



Configurable Environments



Configurable Markov Decision Processes

A **Configurable Markov Decision Process (Conf-MDP)** [Metelli et al., 2018] is an extension of a classic MDP in order to deal with configurable environments.

Configurable Markov Decision Processes

We can think to a Conf-MDP as a system with two entities:

- Learning agent
- Configurator

From a abstract point of view, they act in a **fully-cooperative** scenario.

Configurable Markov Decision Processes

What if the agent and the configurator are no longer cooperative?

Possible scenarios

Supermarket

Possible scenarios

Computer Security

Non-Cooperative Configurable Markov Decision Processes

A **Non-Cooperative Configurable Markov Decision Process (NConf-MDP)** is an extension of Conf-MDP in order to model a non-cooperative interaction between the agent and the configurator.

Markov Decision Processes

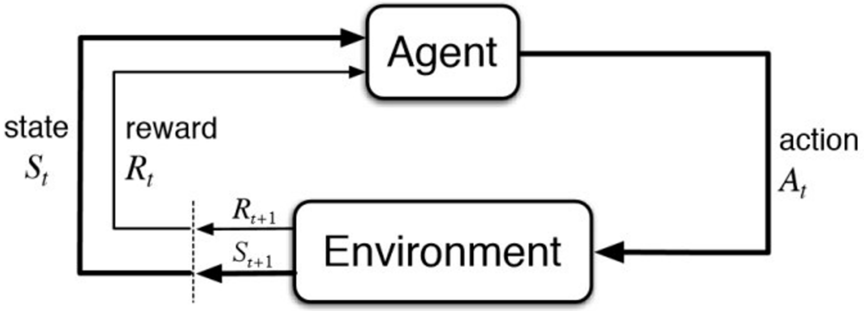
Formally a finite-horizon MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \mu, H)$, where:

- $\mathcal{S}$ is the set of states
- $\mathcal{A}$ is the set of actions
- $p(s'|s, a)$ is the transition model
- $r(s)$ is the reward function
- $\mu(s)$ is the probability distribution over the initial state
- H is the time horizon

Policy

The agent selects actions following a **policy.**

Deterministic policy $\pi = (\pi_1, \pi_2, \ldots, \pi_H)$ in the finite-horizon setting is a sequence of decision rules $\pi_h : \mathcal{S} \to \mathcal{A}$.



Solving a MDP

Solving a MDP means finding the **optimal policy**, i.e. the policy that maximizes the agent performance.

$$\pi^\star = \operatorname*{argmax}_{\pi \in \Pi} V^\pi$$

where $V^\pi = \mathbb{E}\left[\sum_{h=1}^{H} r_h\right]$ is the expected agent's return.

Optimal Value Functions

- **State value function** $V_h^\star : \mathcal{S} \to \mathbb{R}$
  - Represents the value of a state in a time instant h under the optimal policy.
- **State-action value function** $Q_h^\star : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
  - Represents the value of a state-action pair in a time instant h under the optimal policy.

State Value Function

The **state value function** can be defined by this formula, named *Bellman optimality equation***:**

$$V_h^\star(s) = r(s) + \max_{a \in \mathcal{A}} \left[\sum_{s'} p(s'|s,a) V_{h+1}^\star(s') \right]$$

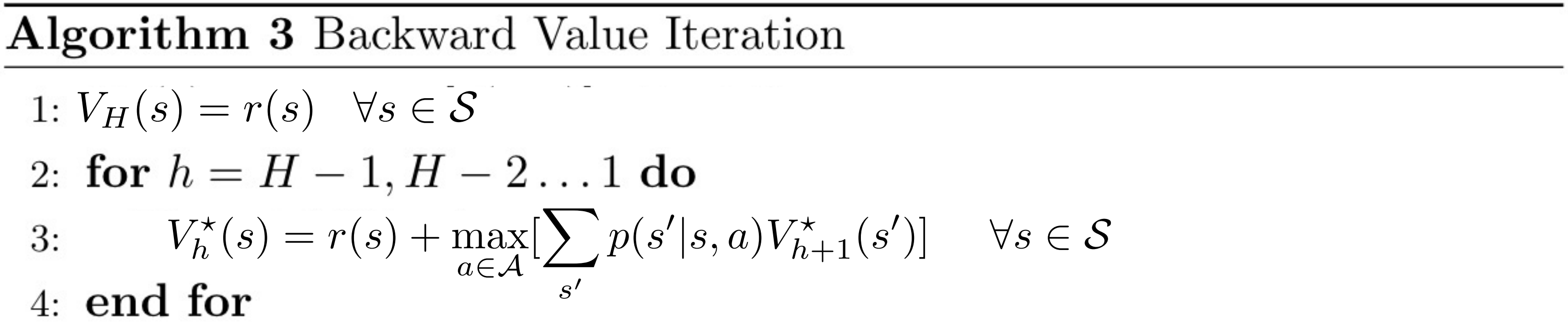
$$\Big(\quad V_H^\star(s) = r(s) \quad \Big)$$

State-action Value Function

The **state-action value function** can be derived starting from the value function:

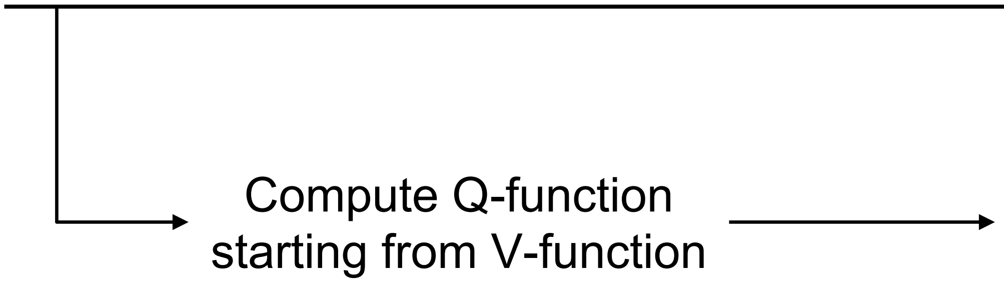
$$Q^\star_h(s,a) = r(s) + \sum_{s'} p(s'|s,a) V^\star_{h+1}(s')$$

Backward Value Iteration



Algorithm 3 Backward Value Iteration

```
$V_H(s) = r(s) \quad \forall s \in \mathcal{S}$
for $h = H-1, H-2 \dots 1$ do
    $V^\star_h(s) = r(s) + \max_{a \in \mathcal{A}} [\sum_{s'} p(s'|s,a) V^\star_{h+1}(s')] \quad \forall s \in \mathcal{S}$
end for
```



Compute Q-function starting from V-function

Compute the greedy policy

$$\pi_h(s) \leftarrow \text{argmax}_a [Q_h(s,a)] \quad \forall h \in [H]$$

Configurable Markov Decision Processes

Formally, a **finite-horizon Conf-MDP** is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mu, H)$, where:

- $(\mathcal{S}, \mathcal{A}, r, \mu, H)$ is a finite-horizon MDP without the transition model
- $\mathcal{P}$ is the *set of transition models*

Objective: Find the model-policy pair (p, π) that maximize the agent performance.

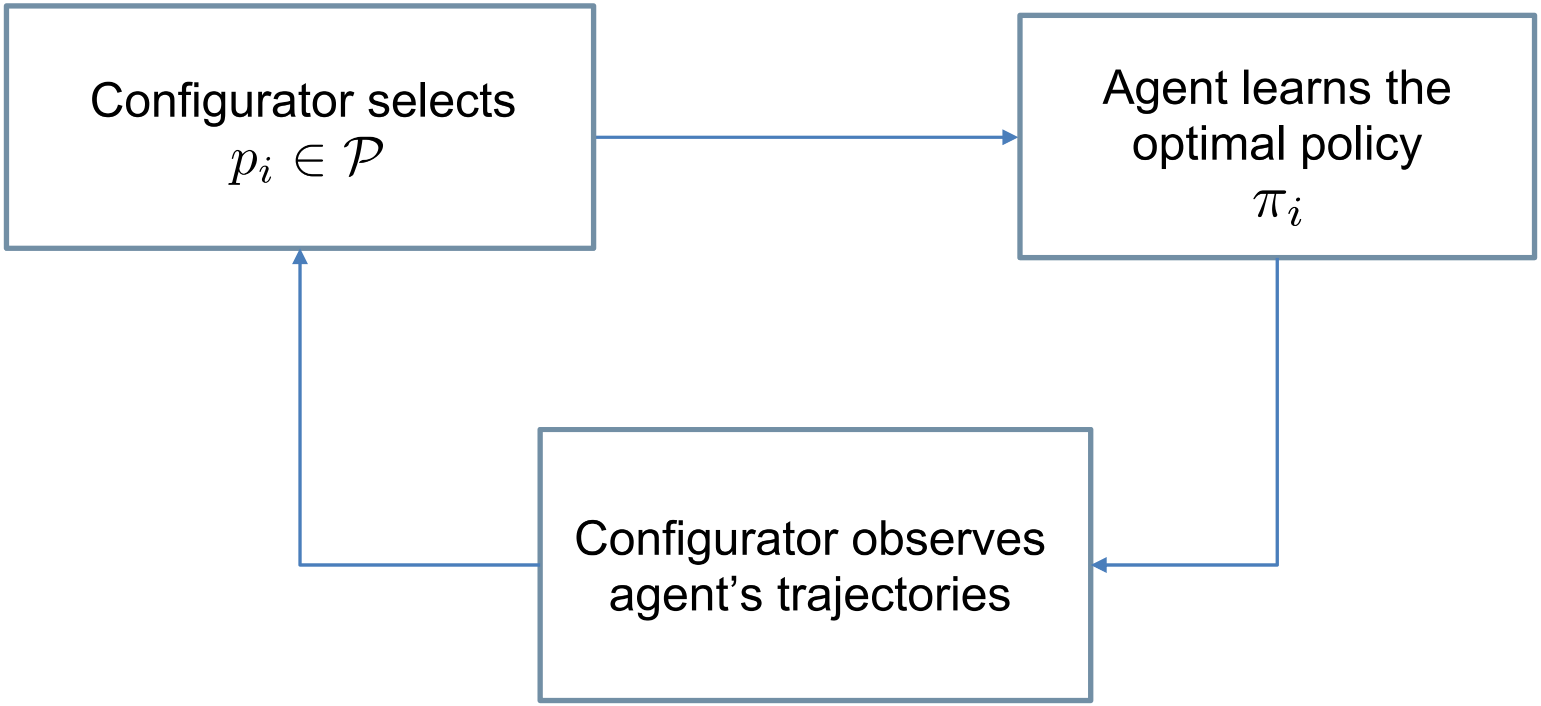
Non-Cooperative Configurable Markov Decision Processes

Formally, a NConf-MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r_o, r_c, \mu, H)$, where:

- $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, H)$ is a finite-horizon Conf-MDP without the reward function
- $r_o(s)$ is the reward function of the agent (opponent)
- $r_c(s)$ is the reward function of the configurator

Objective: Find the model-policy pair (p, π) that maximize the configurator performance, knowing that π is optimal in p .

Non-Cooperative Configurable Markov Decision Processes



Configurator selects $p_i \in \mathcal{P}$

Agent learns the optimal policy π_i

Configurator observes agent’s trajectories

Non-Cooperative Configurable Markov Decision Problem

From a game-theoretic point of view, the interaction between the agent and the configuration can be modelled using **Stackelberg Games**.

Stackelberg Games

The simplest formulation of Stackelberg game is characterized by two players, a **leader** and a **follower**, that interact in a hierarchical structure:

1. The leader plays its strategy first.
2. The follower plays its best response

Stackelberg Games

The leader aims to solve this optimization problem:

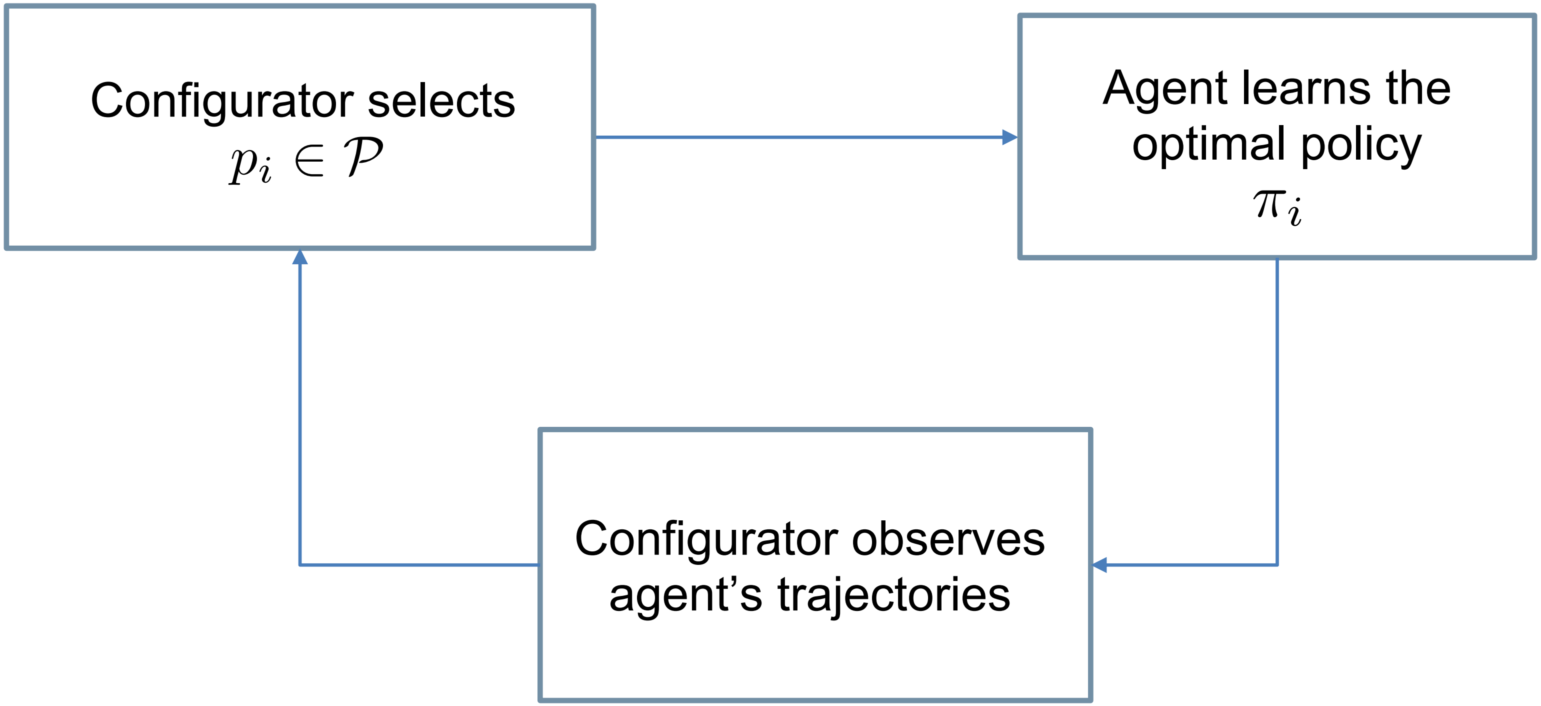
$$\max_{a_1 \in \mathcal{A}} \{ r_1(a_1, BR(a_1)) \}$$

where $BR(a_1) \in \arg\max_{a \in \mathcal{A}_2} r_2(a_1, a)$.

While the follower aims to solve this optimization problem:

$$\max_{a_2 \in \mathcal{A}_2} r_2(a_1, a_2).$$

Non-Cooperative Configurable Markov Decision Processes

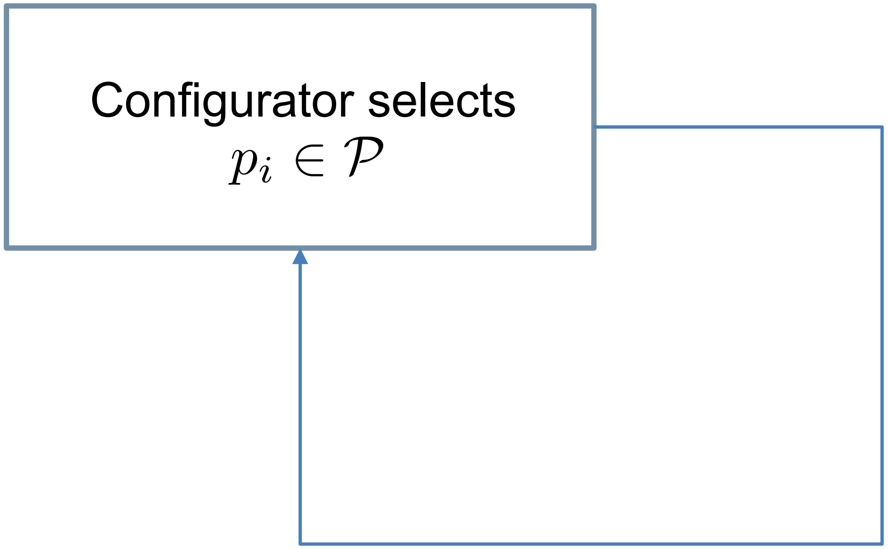


Configurator selects $p_i \in \mathcal{P}$

Agent learns the optimal policy π_i

Configurator observes agent's trajectories

Non-Cooperative Configurable Markov Decision Processes



Configurator observes its performance

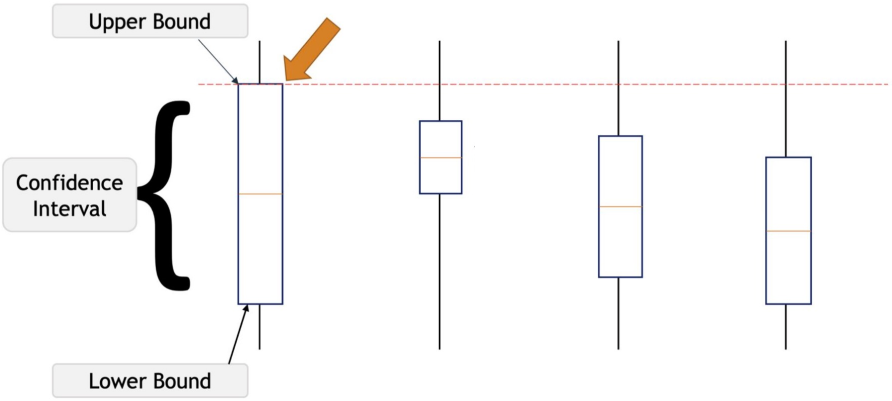
Non-Cooperative Configurable Markov Decision Processes

If we ignore the structure of the problem we could cast the problem of learning the best configuration to a **Multi-armed Bandit**.

Upper Confidence Bound

Multi-armed Bandits are a special class of MDPs with only one state.

Upper Confidence Bound (UCB) solve MAB problem using the ”Optimism in Face of Uncertainty” (OFU) principle.



Performance of MAB algorithms

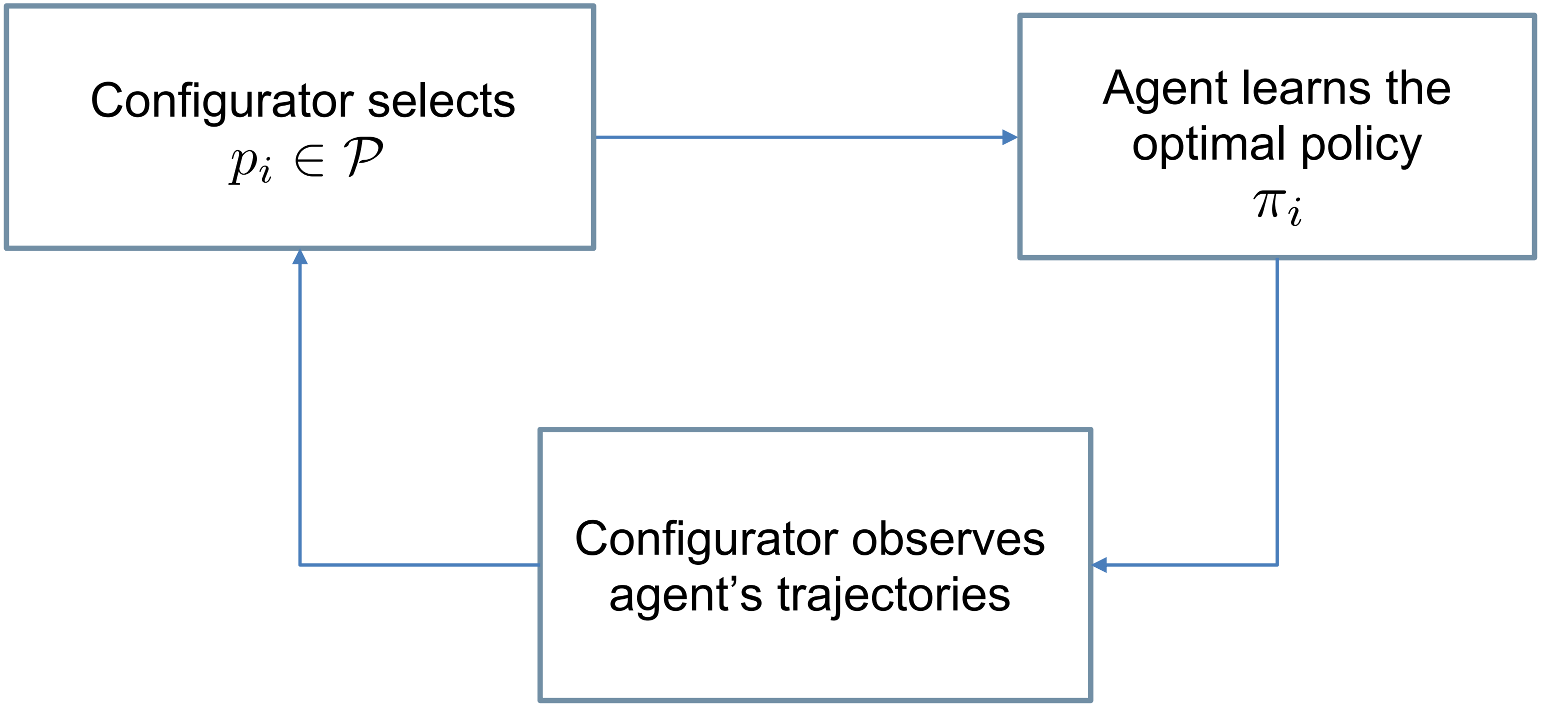
We can measure the performance of a generic MAB algorithm using the **regret:**

$$\Delta = \mathbb{E}\left[\sum_{k=1}^{K} \max_{a \in \mathcal{A}} V_a - V_{a_k}\right]$$

Value of the best action

Value of the action performed in episode k

Non-Cooperative Configurable Markov Decision Processes



Configurator selects
$p_i \in \mathcal{P}$

Agent learns the optimal policy
π_i

Configurator observes agent's trajectories

Non-Cooperative Configurable Markov Decision Processes

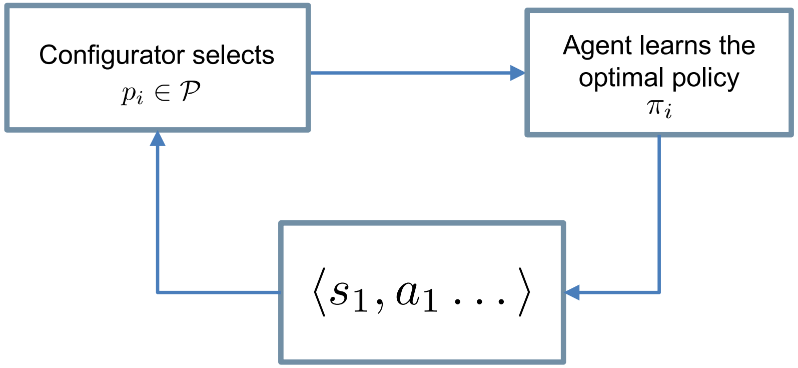
We propose two algorithms for solving NConf-MDPs:

- Action-feedback Optimistic Configuration Learning (AfOCL)
- Reward-feedback Optimistic Configuration Learning (RfOCL)

Non-Cooperative Configurable Markov Decision Processes

We study two different types of feedback:

Action-feedback

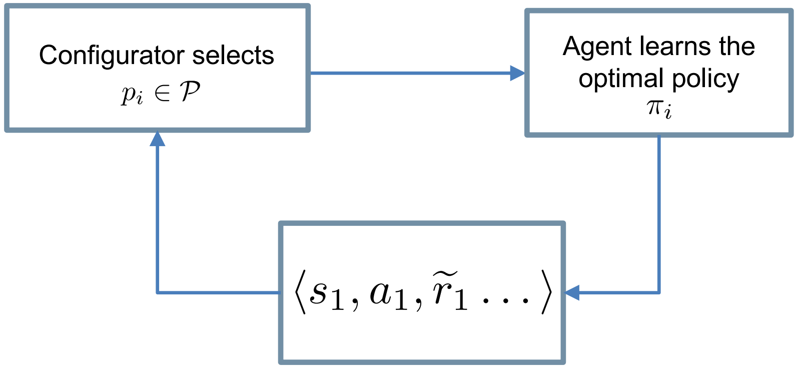


Configurator selects $p_i \in \mathcal{P}$

Agent learns the optimal policy π_i

$\langle s_1, a_1 \dots \rangle$

Reward-feedback



Configurator selects $p_i \in \mathcal{P}$

Agent learns the optimal policy π_i

$\langle s_1, a_1, \widetilde{r}_1 \dots \rangle$

Action-feedback Optimistic Configuration Learning

Trajectories are composed by states and actions only…

$$\langle s_1, a_1, s_2, a_2, \ldots, s_{H-1}, a_{H-1}, s_H \rangle$$

where $a_h = \pi_{i,h}(s_h)$.

Assumption 1:

The agent’s policy is deterministic and fixed.

… but the transition model is stochastic!

Action-feedback Optimistic Configuration Learning

AfOCL is based on the **OFU principle**.

Every episode $k \in [K]$ the configurator computes an **optimistic** estimate $\widetilde{V}_k^i$ of its expected return for each configuration $i \in [M]$.

↓

Then, it selects $i \in \arg\max_{i \in [M]} \widetilde{V}_k^i$.

Action-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

We maintain a set of possible policies in each configuration.

↓

We compute $\widetilde{V}_k^i$ using the optimistic policy.

Action-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

From a practical point of view…

Configuration i

$$\mathcal{A}^i_{k,h}(s) \subseteq \mathcal{A}$$

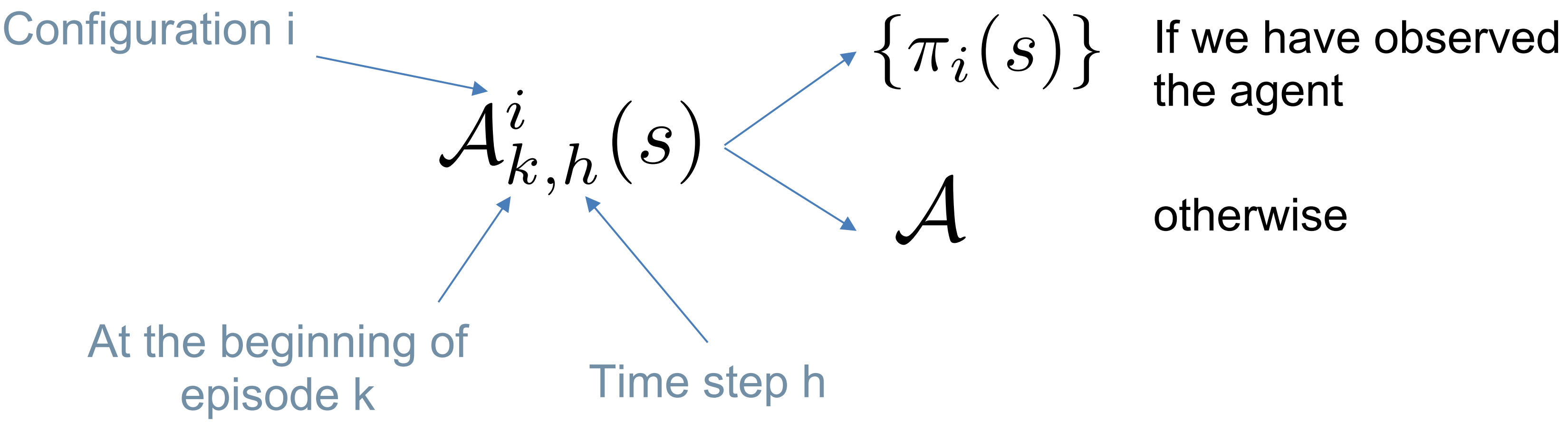
At the beginning of episode k

Time step h

Action-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

From a practical point of view…



Configuration i

$\mathcal{A}_{k,h}^{i}(s)$

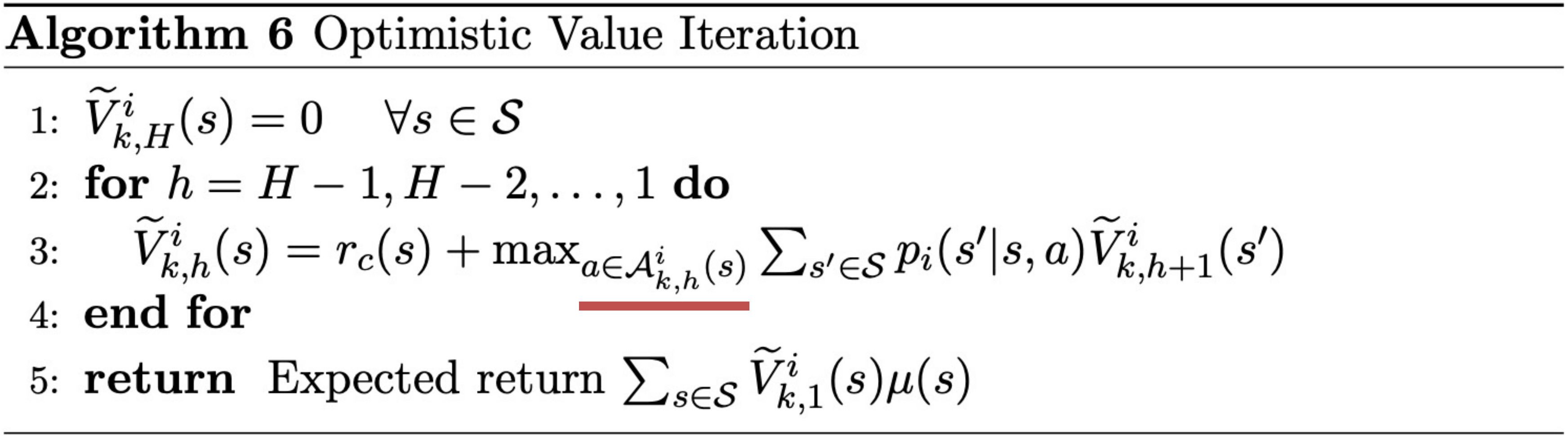
$\{\pi_i(s)\}$ If we have observed the agent

$\mathcal{A}$ otherwise

At the beginning of episode k

Time step h

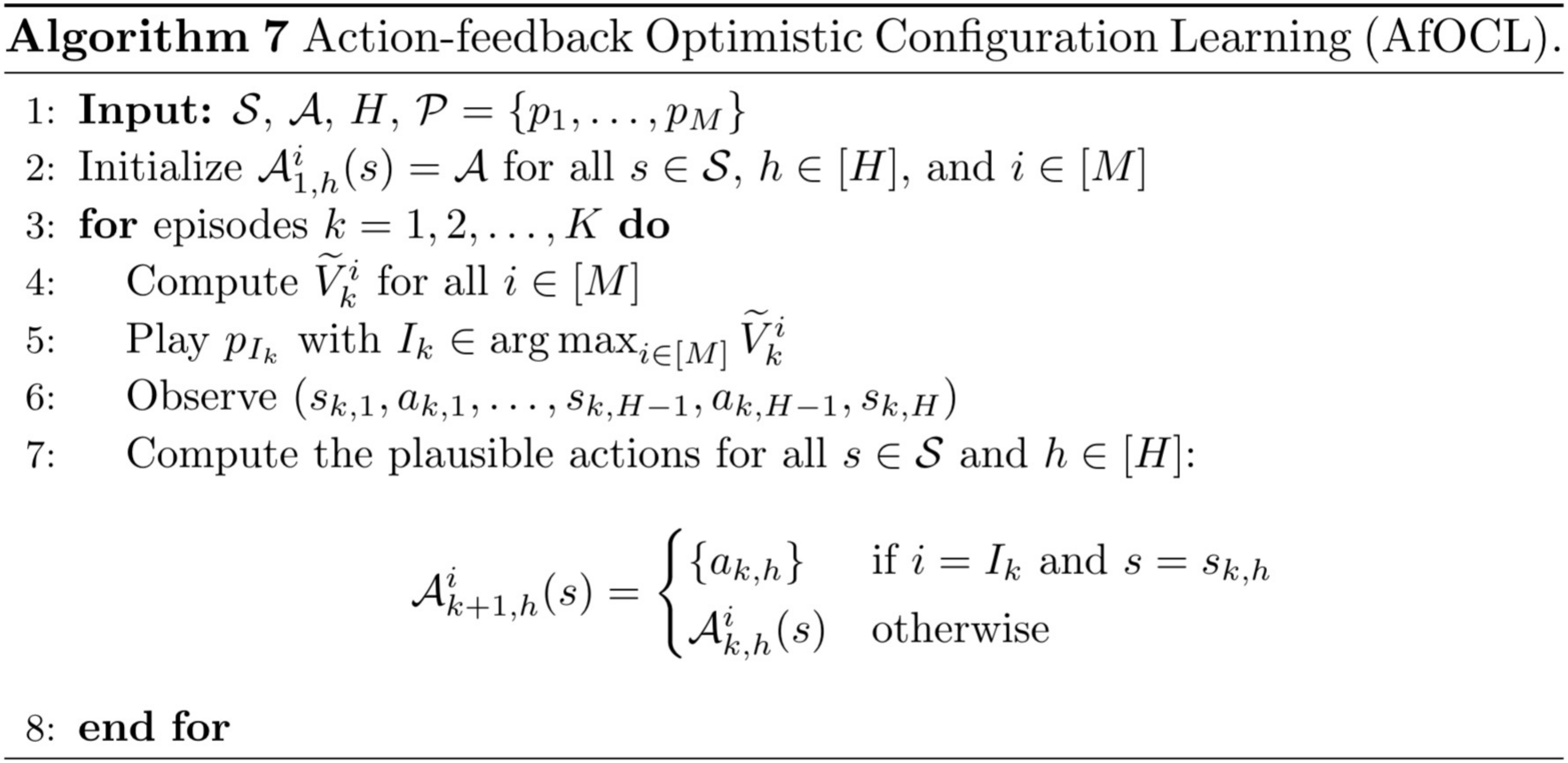
Action-feedback Optimistic Configuration Learning



Algorithm 6 Optimistic Value Iteration

```
$\widetilde{V}^i_{k,H}(s) = 0 \quad \forall s \in \mathcal{S}$
for $h = H-1, H-2, \dots, 1$ do
    $\widetilde{V}^i_{k,h}(s) = r_c(s) + \max_{\underline{a \in \mathcal{A}^i_{k,h}(s)}} \sum_{s' \in \mathcal{S}} p_i(s'|s,a)\widetilde{V}^i_{k,h+1}(s')$
end for
return Expected return $\sum_{s \in \mathcal{S}} \widetilde{V}^i_{k,1}(s)\mu(s)$
```

Action-feedback Optimistic Configuration Learning



Algorithm 7 Action-feedback Optimistic Configuration Learning (AfOCL).

```
**Input:** $\mathcal{S}$, $\mathcal{A}$, $H$, $\mathcal{P} = \{p_1, \dots, p_M\}$
Initialize $\mathcal{A}^i_{1,h}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$
**for** episodes $k = 1, 2, \dots, K$ **do**
    Compute $\widetilde{V}^i_k$ for all $i \in [M]$
    Play $p_{I_k}$ with $I_k \in \arg\max_{i \in [M]} \widetilde{V}^i_k$
    Observe $(s_{k,1}, a_{k,1}, \dots, s_{k,H-1}, a_{k,H-1}, s_{k,H})$
    Compute the plausible actions for all $s \in \mathcal{S}$ and $h \in [H]$:
        $$\mathcal{A}^i_{k+1,h}(s) = \begin{cases} \{a_{k,h}\} & \text{if } i = I_k \text{ and } s = s_{k,h} \\ \mathcal{A}^i_{k,h}(s) & \text{otherwise} \end{cases}$$
**end for**
```

Action-feedback Optimistic Configuration Learning

Regret guarantees

Under Assumption 1, the expected regret of AfOCL at every episode K is bounded by:

$$\mathbb{E}[Regret(K)] \leq MH^3S^2.$$

The upper bound does not depend on the number of episodes K!

Action-feedback Optimistic Configuration Learning

There is no way to transfer information across different configurations!

Reward-feedback Optimistic Configuration Learning

Trajectories are composed by states, actions and a noisy version of rewards…

$$\langle s_1, a_1, \widetilde{r}_1, \ldots s_{H-1}, a_{H-1}, \widetilde{r}_{H-1}, s_H \rangle$$

Assumption 2:

The MDP induced by the best response policy must be *ergotic*.

Reward-feedback Optimistic Configuration Learning

RfOCL is able to **transfer knowledge** across different configurations using an estimate of the reward function of the agent.

Reward-feedback Optimistic Configuration Learning

RfOCL and AfOCL share the **same structure**.

Every episode $k \in [K]$ the configurator computes an **optimistic** estimate $\widetilde{V}_k^i$ of its expected return for each configuration $i \in [M]$.

Then, it selects $i \in \arg\max_{i \in [M]} \widetilde{V}_k^i$.

Reward-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

We still maintain a set of plausible actions:

$$\begin{tikzcd}
 & \{\pi_i(s)\} & \text{If we have observed the agent} \\
\mathcal{A}^i_{k,h}(s) \arrow[ur] \arrow[dr] & & \\
 & ? & \text{otherwise}
\end{tikzcd}$$

Reward-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

1. We compute a confidence interval $\mathcal{R}_k(s) = [\underline{r}_{o,k}(s), \overline{r}_{o,k}(s)]$ of the **agent**'s reward function using Hoeffding's inequality:

$$\widehat{r}_{o,k}(s) \pm \sqrt{\frac{\log(SHk^3)}{\max\{N_k(s), 1\}}}$$

Reward-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}^i_k$?

2. Compute the confidence interval on the Q functions of the **agent** induced by $\mathcal{R}_k(s)$ in each configurations.

$$\mathcal{Q}^i_{k,h}(s,a) = [\underline{Q}^i_{o,k,h}(s,a), \overline{Q}^i_{o,k,h}(s,a)]$$

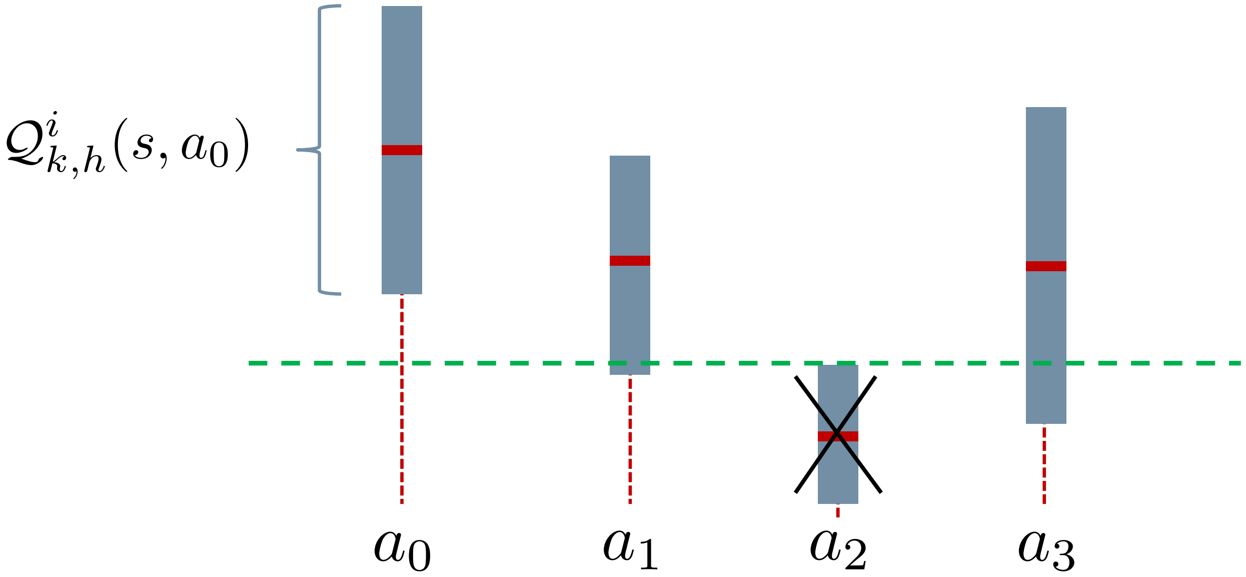
Value iteration with $\underline{r}_{o,k}(s)$

Value iteration with $\overline{r}_{o,k}(s)$

Reward-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

3. We discard actions that are ”dominated” by other actions



Reward-feedback Optimistic Configuration Learning

How to compute the optimistic expected return $\widetilde{V}_k^i$?

3. We discard actions that are ”dominated” by other actions

$$\widetilde{\mathcal{A}}^i_{k,h}(s) = \left\{ a \in \mathcal{A} : \overline{Q}^i_{o,k,h}(s,a) \geq \max_{a' \in \mathcal{A}} \underline{Q}^i_{o,k,h}(s,a') \right\}$$

Reward-feedback Optimistic Configuration Learning

Regret guarantees

Under Assumption 2, the expected regret of RfOCL at every episode K is bounded by:

$$\mathbb{E}[Regret(K)] \leq \overline{K}\Delta + \frac{\pi^2}{3}$$

Constant depending on S and H

Maximum suboptimality gap

The upper bound does not depend on the number of configuration M!

Experimental Evaluation

What we want to show:

1. AfOCL and RfOCL bring advantages over a MAB approach (UCB1).
2. RfOCL performs better than AfOCL if Assumption 2 holds.
3. RfOCL is able to scale very well with a high number of configurations.

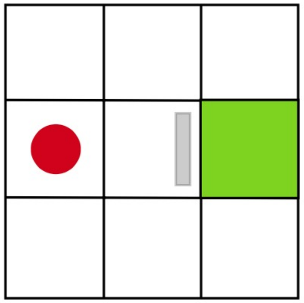
Experimental Evaluation

We compare our algorithms with UCB in three different domains:

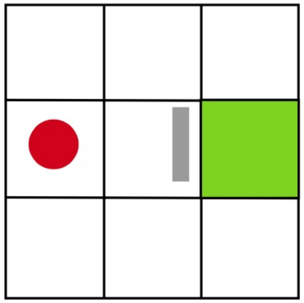
- Configurable Gridworld
- Teacher-Student
- Marketplace

Experimental Evaluation

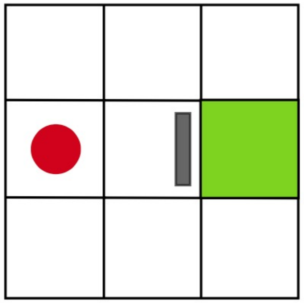
Configurable Gridworld



Configuration #1



Configuration #2

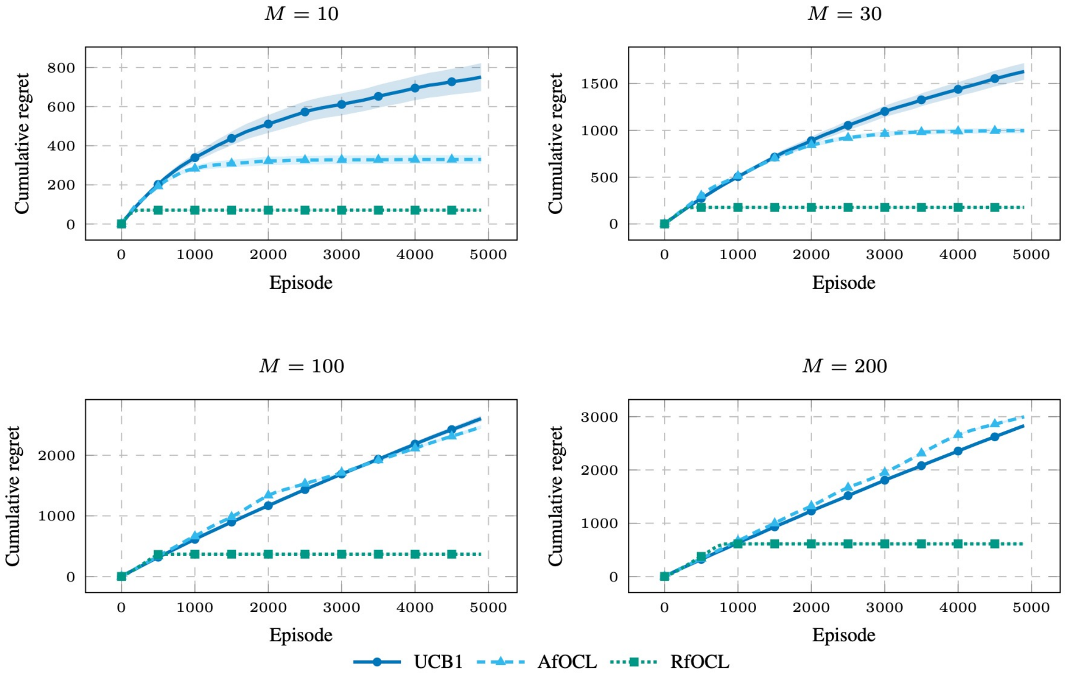


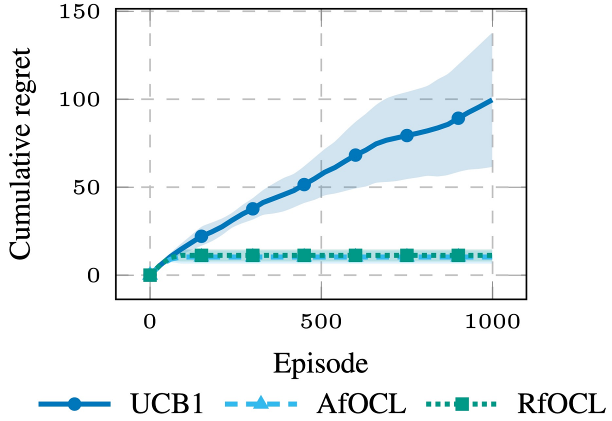
Configuration #3

- The agent's goal is to reach the terminal state as soon as possible.
- The configurator’s goal is to keep the agent in the central cell as long as possible.

Experimental Evaluation

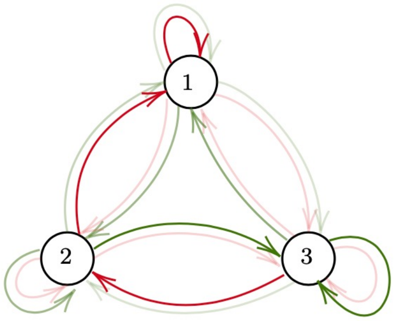
Configurable Gridworld – Experiment



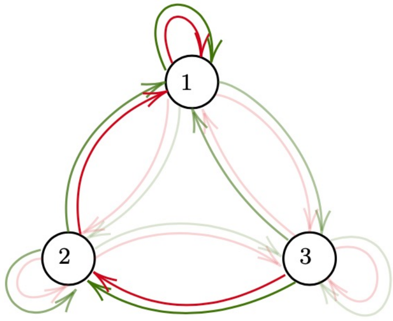


Experimental Evaluation

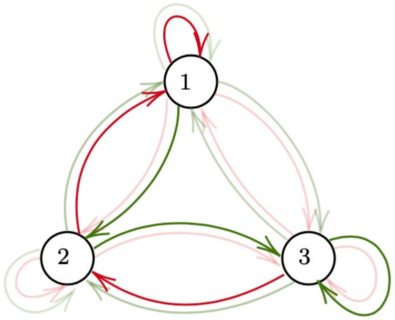
Student-Teacher



Configuration #1



Configuration #2

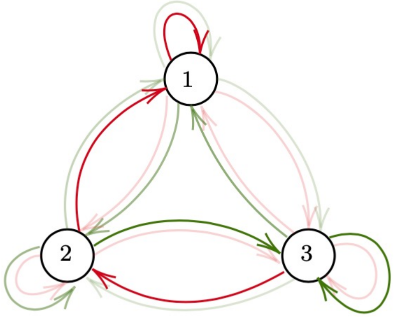


Configuration #3

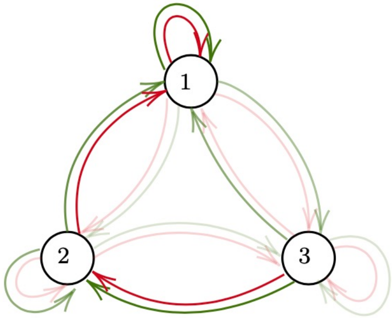
- The teacher (configurator) has a list of S exercises characterized by a different level of difficulty.
- The goal of the teacher is to find the right sequence of exercises.

Experimental Evaluation

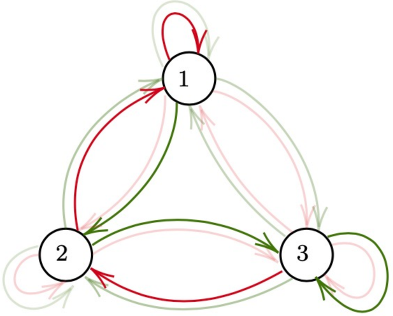
Student-Teacher



Configuration #1



Configuration #2

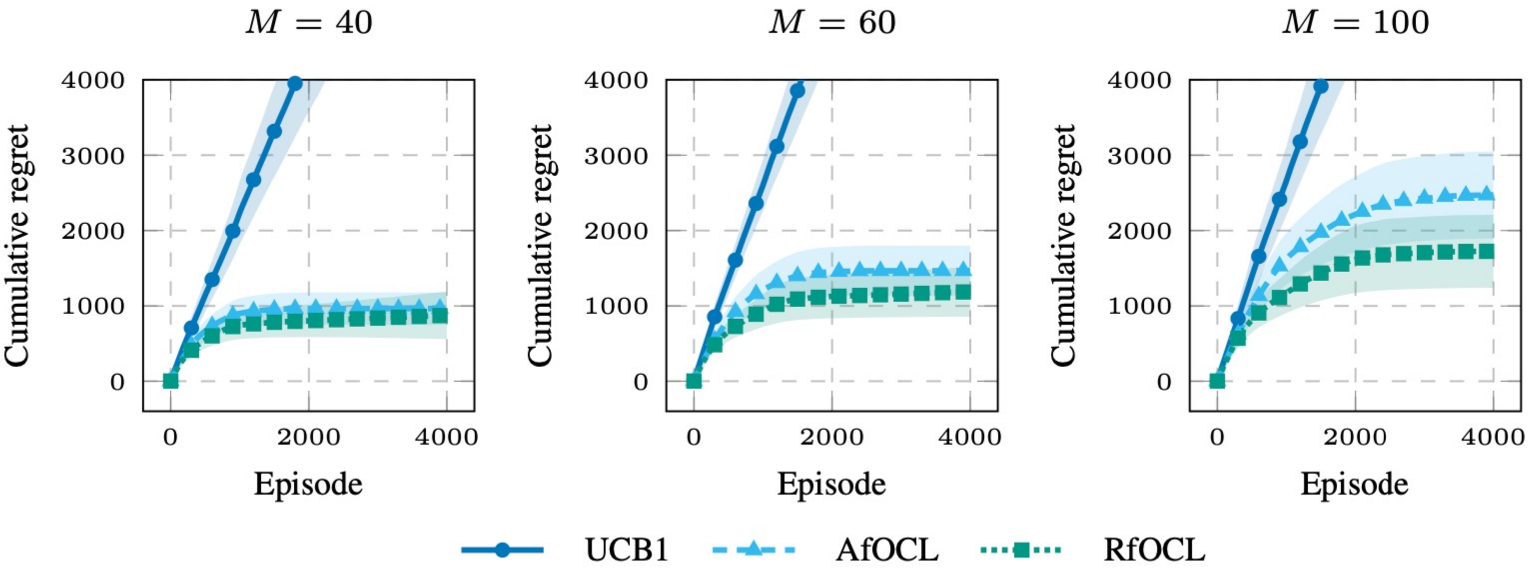


Configuration #3

- The student (agent) perceives the level of difficulties of the exercises in a different way and it can decide to not answer the ones he find too difficult.
- The goal of the student is the same of the teacher: start solving most difficult exercises as soon as possible!

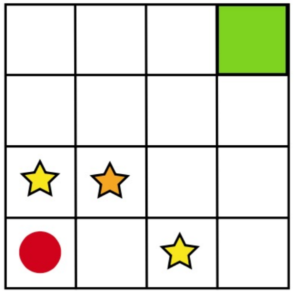
Experimental Evaluation

Student-Teacher - Experiment

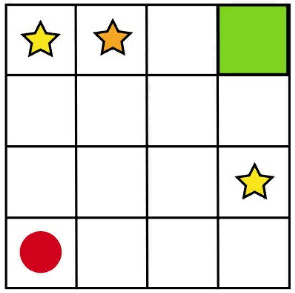


Experimental Evaluation

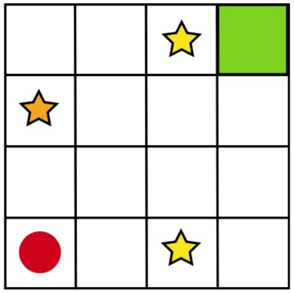
Marketplace



Configuration #1



Configuration #2

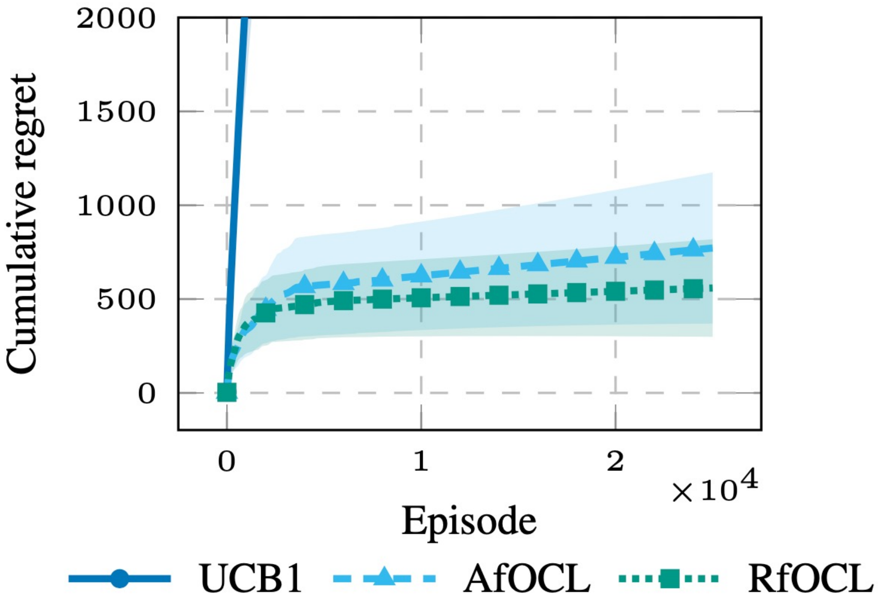


Configuration #3

- The customer's goal is grab the only product it is interested in and reach the exit.
- The goal of the supermarket owner is to induce the customer to buy other products.

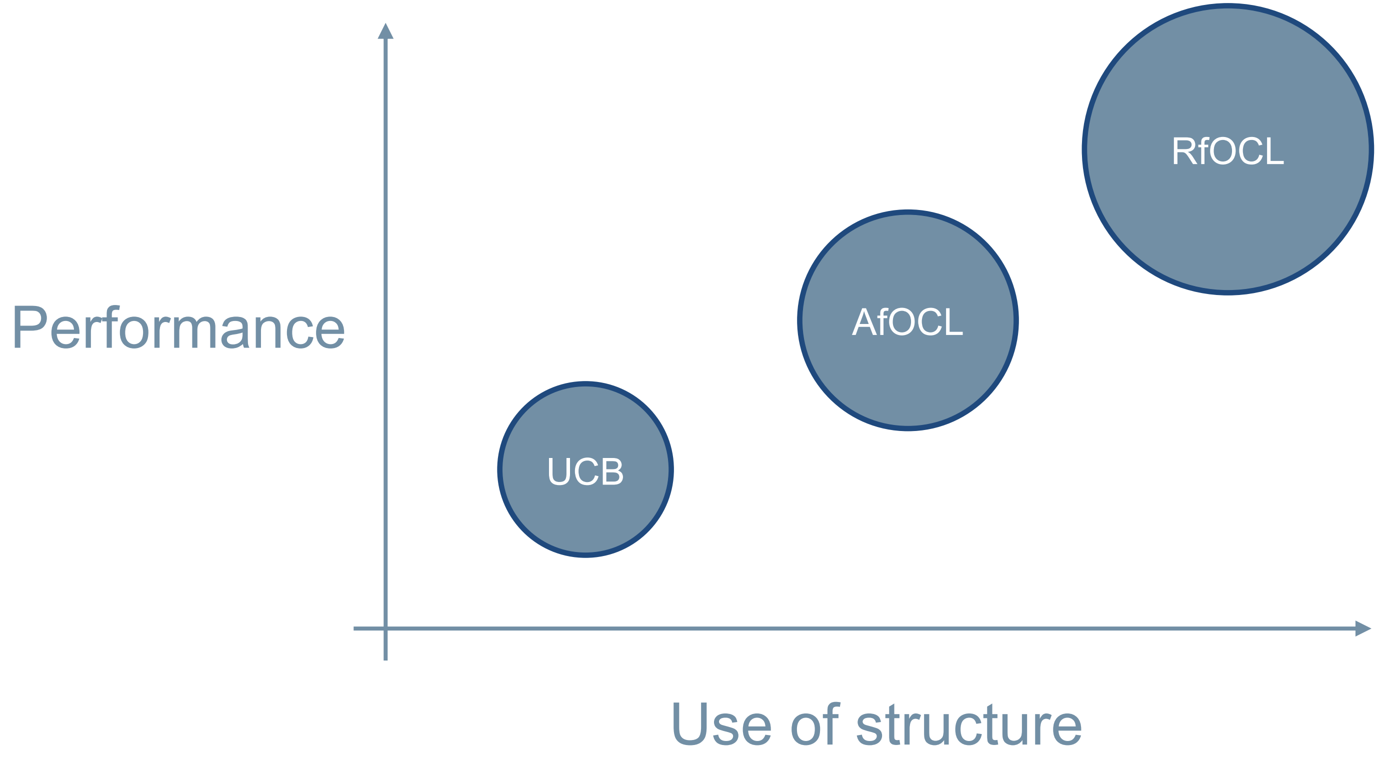
Experimental Evaluation

Marketplace - Experiment



Conclusions

Solving Non Cooperative Conf-MDPs



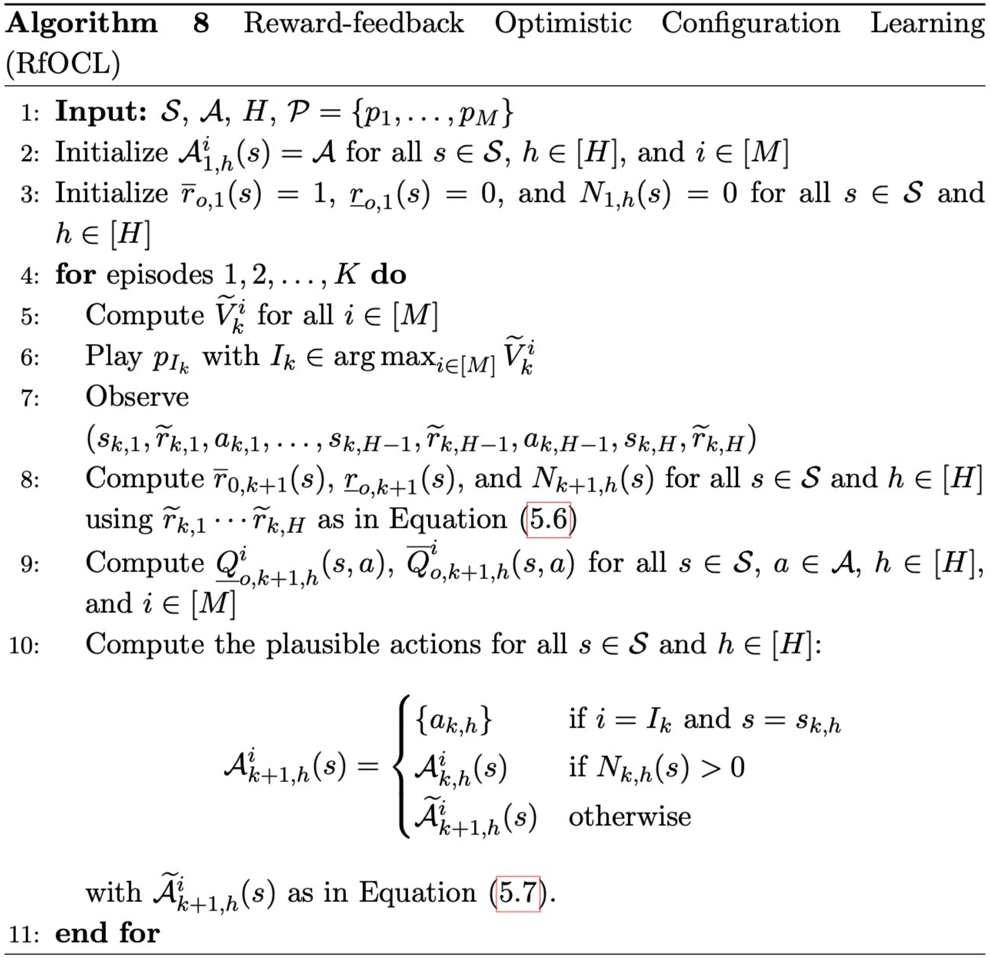
Future Research Directions

- Fixed Stochastic policy
- Awareness of the agent
- Inverse Reinforcement Learning

Thanks for your attention!

Alessandro Concetti

Reward-feedback Optimistic Configuration Learning



Algorithm 8 Reward-feedback Optimistic Configuration Learning (RfOCL)

```
**Input:** $\mathcal{S}$, $\mathcal{A}$, $H$, $\mathcal{P} = \{p_1, \dots, p_M\}$
Initialize $\mathcal{A}^i_{1,h}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$
Initialize $\overline{r}_{o,1}(s) = 1$, $\underline{r}_{o,1}(s) = 0$, and $N_{1,h}(s) = 0$ for all $s \in \mathcal{S}$ and $h \in [H]$
**for** episodes $1, 2, \dots, K$ **do**
    Compute $\widetilde{V}^i_k$ for all $i \in [M]$
    Play $p_{I_k}$ with $I_k \in \arg\max_{i \in [M]} \widetilde{V}^i_k$
    Observe $(s_{k,1}, \widetilde{r}_{k,1}, a_{k,1}, \dots, s_{k,H-1}, \widetilde{r}_{k,H-1}, a_{k,H-1}, s_{k,H}, \widetilde{r}_{k,H})$
    Compute $\overline{r}_{0,k+1}(s)$, $\underline{r}_{o,k+1}(s)$, and $N_{k+1,h}(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$ using $\widetilde{r}_{k,1} \cdots \widetilde{r}_{k,H}$ as in Equation (5.6)
    Compute $\underline{Q}^i_{o,k+1,h}(s,a)$, $\overline{Q}^i_{o,k+1,h}(s,a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $h \in [H]$, and $i \in [M]$
    Compute the plausible actions for all $s \in \mathcal{S}$ and $h \in [H]$:
        $$\mathcal{A}^i_{k+1,h}(s) = \begin{cases} \{a_{k,h}\} & \text{if } i = I_k \text{ and } s = s_{k,h} \\ \mathcal{A}^i_{k,h}(s) & \text{if } N_{k,h}(s) > 0 \\ \widetilde{\mathcal{A}}^i_{k+1,h}(s) & \text{otherwise} \end{cases}$$
    with $\widetilde{\mathcal{A}}^i_{k+1,h}(s)$ as in Equation (5.7).
**end for**
```

Experimental Evaluation

Marketplace

- **Number of states:** 16
- **Number of actions:** 4
- **Agent's reward:** -1 everywhere and 0.9 where there is the product.
- **Configurator's reward:** 0 everywhere and 1 where there is some products.
- **Configurations:** M random transition models

Experimental Evaluation

Student-Teacher – Nconf-MDP

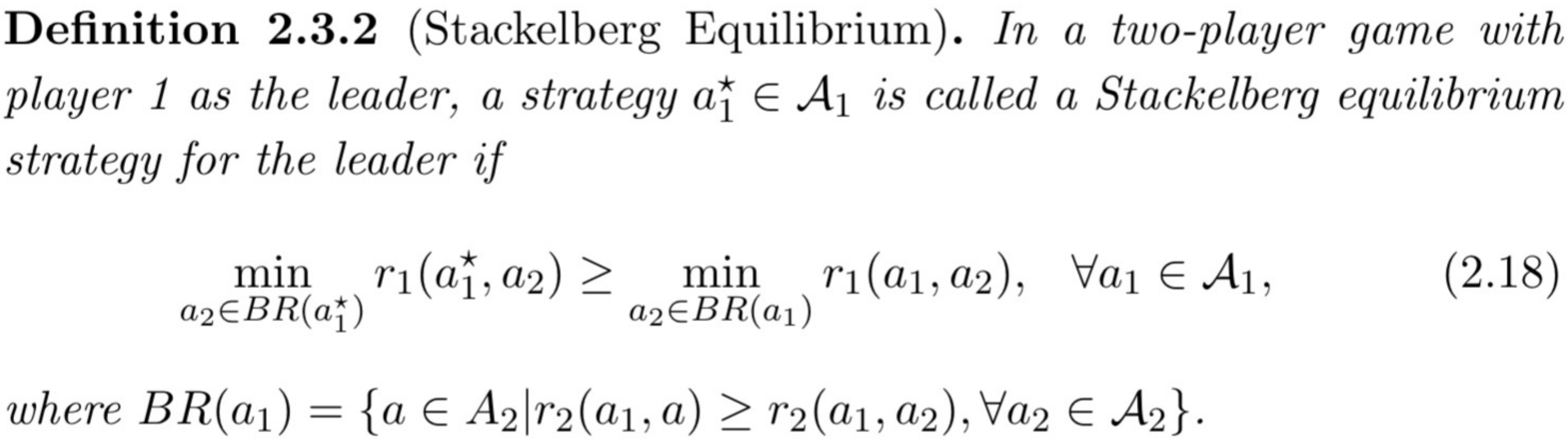
- **Number of states:** 10 (exercises)
- **Number of actions:** 2 (answer/not answer)
- **Agent's reward:** difficulty perceived by the agent
- **Configurator's reward:** difficulty perceived by the configurator
- **Configurations:** M transition models that differ each other by the way they assign the probabilities to next states when the agent decides to answer.

Experimental Evaluation

Configurable Gridworld – Nconf-MDP

- **Number of states:** 9
- **Number of actions:** 4
- **Agent’s reward:** -1 everywhere
- **Configurator’s reward:** 0 everywhere and 1 in the central cell
- **Configurations:** M transition models with different values of p

Stackelberg Games



Definition 2.3.2 (Stackelberg Equilibrium). *In a two-player game with player 1 as the leader, a strategy* $a_1^\star \in \mathcal{A}_1$ *is called a Stackelberg equilibrium strategy for the leader if*

$$\min_{a_2 \in BR(a_1^\star)} r_1(a_1^\star, a_2) \geq \min_{a_2 \in BR(a_1)} r_1(a_1, a_2), \quad \forall a_1 \in \mathcal{A}_1, \tag{2.18}$$

where $BR(a_1) = \{a \in A_2 | r_2(a_1, a) \geq r_2(a_1, a_2), \forall a_2 \in \mathcal{A}_2\}$.