Model-Based Learning

The other learning technic

Guillaume Lozenguez

@imt-nord-europe.fr

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Model-based learning

Main Idea:

- Random trajectories (a lot)
- Until each transition is visited several times.
- Compute an optimal policy.

Potentially:

- Require driving exploration
- Only incomplete exploration can be performed2

But first the Model

Markov Descision Process

A framework for modeling stockastic evolution of system to control.

Bellman equation

Recurciv evaluation of states to compute exepected gains.

Solving algorythms

- Value iteration
- Policy iteration

Markov Decision Process

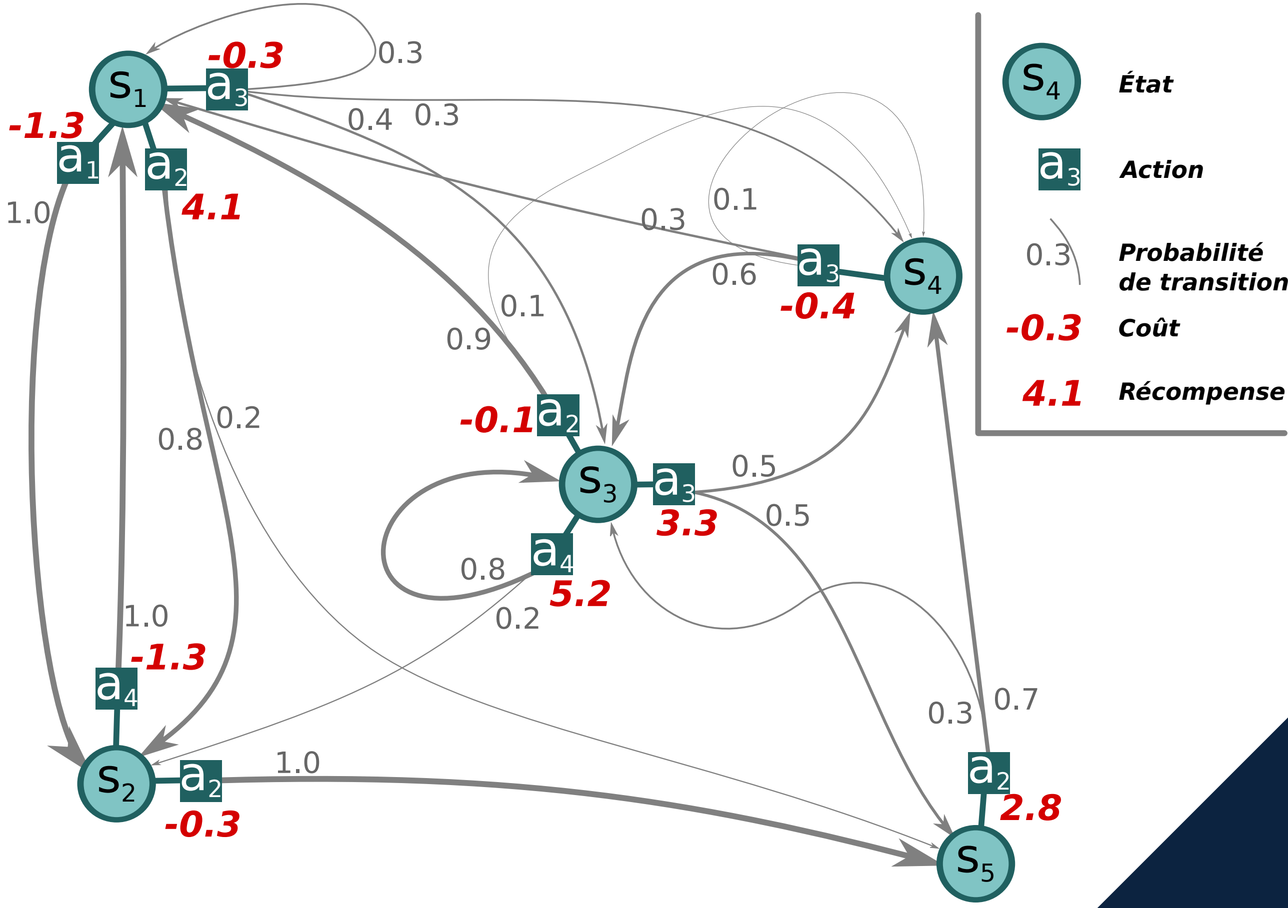
MDP: $\langle S, A, T, R \rangle$:

S : set of system's states
A : set of possible actions
T : S × A × S → [0, 1] : transitions
R : S × A → R : cost/rewards

Optimal policy:

π: a function returning the action to perform in each crossed states.

π^*: the optimal policy maximizing the gains (expected cumulated rewards).



Choosing : building a policy of action

Example of policy in 421:

π^{421}: Always target a 4-2-1 (keep only one **4**, one **2** and one **1**).

s	$\pi^{421}(s)$	s	$\pi^{421}(s)$
h-*1*-1-1	*keep*-roll-roll	…	
h-*2*-*1*-1	*keep*-*keep*-roll	h-*4*-*2*-*1*	*keep*-*keep*-*keep*
h-3-*1*-1	roll-*keep*-roll	…	
h-*4*-*1*-1	*keep*-*keep*-roll	h-6-6-5	roll-roll-roll
…		h-6-6-6	roll-roll-roll

(Invariant over the horizon h)

Bellman Equation

State evaluation for a given policy π :

$$V^\pi(s) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \times V^\pi(s')$$

$$\text{with} : a = \pi(s) \text{ and } \gamma \in [0,1] \text{ the discount factor (typically } 0.99)$$

As a sum of gains:

- The immediate reward: $R(s,a)$.
- The future gains $V^\pi(s')$, for all possible next states $s' \in S$,
- proportionally to the probability to reach them $T(s,a,s')$

Solving MDP: Value Iteration

Input: an **MDP:** $\langle S, A, T, R\rangle$; precision error: ϵ ; discount factor: γ ; initial **V(s)**

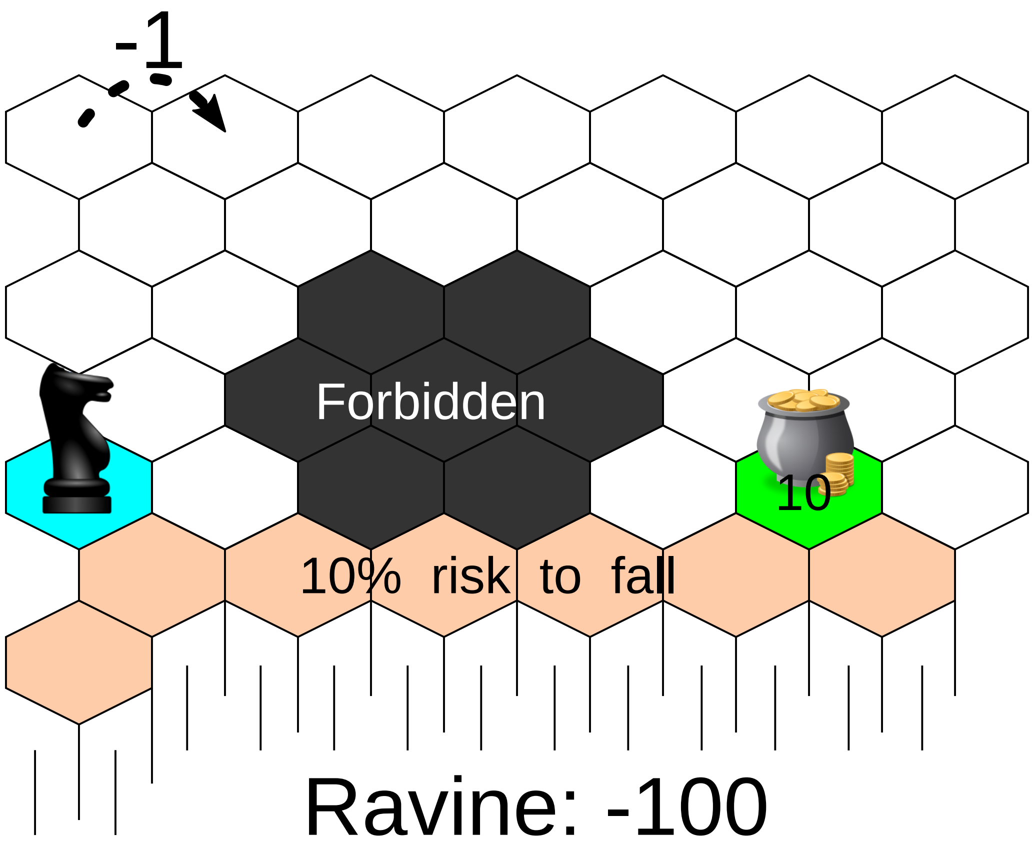
1. Repeat until the **maximal delta* $< \epsilon$

   For each state $s \in S$

   - Search the action a^* maximizing the Bellman Equation on s
   - Update $\pi(s)$ and **V()** by considering action a^*
   - Compute the delta value between the previous and the new **V(S)**

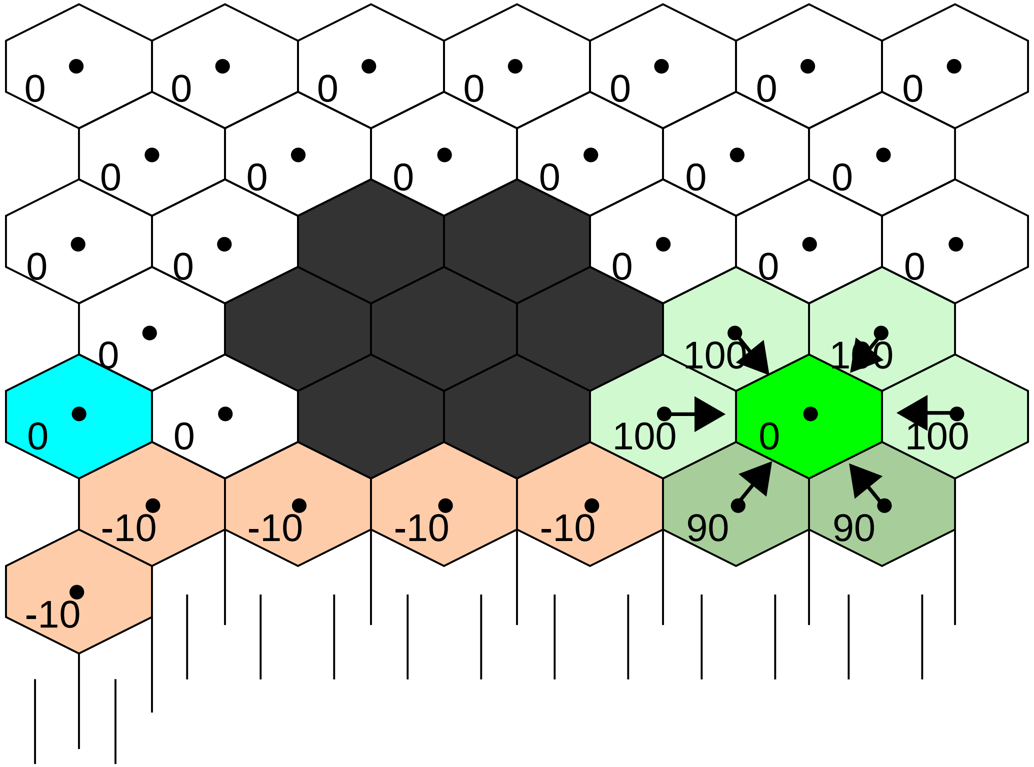
Output: an optimal π^* and associated V-values

Example: Dangerous move.



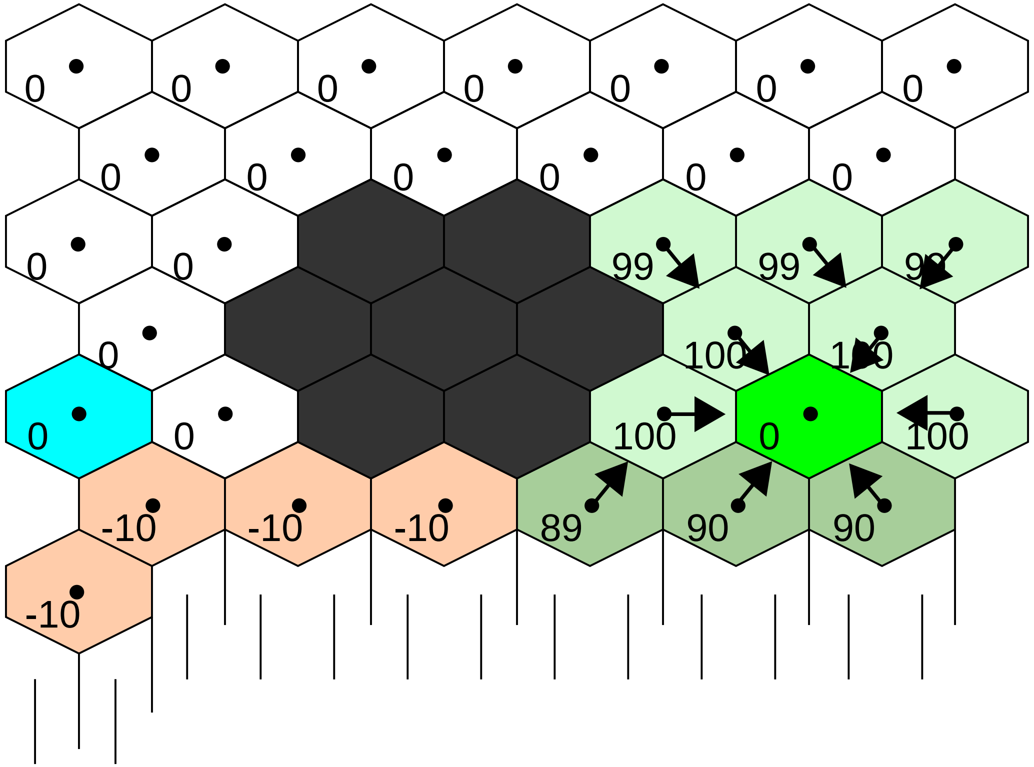
Probleme definition

Example: Dangerous move.



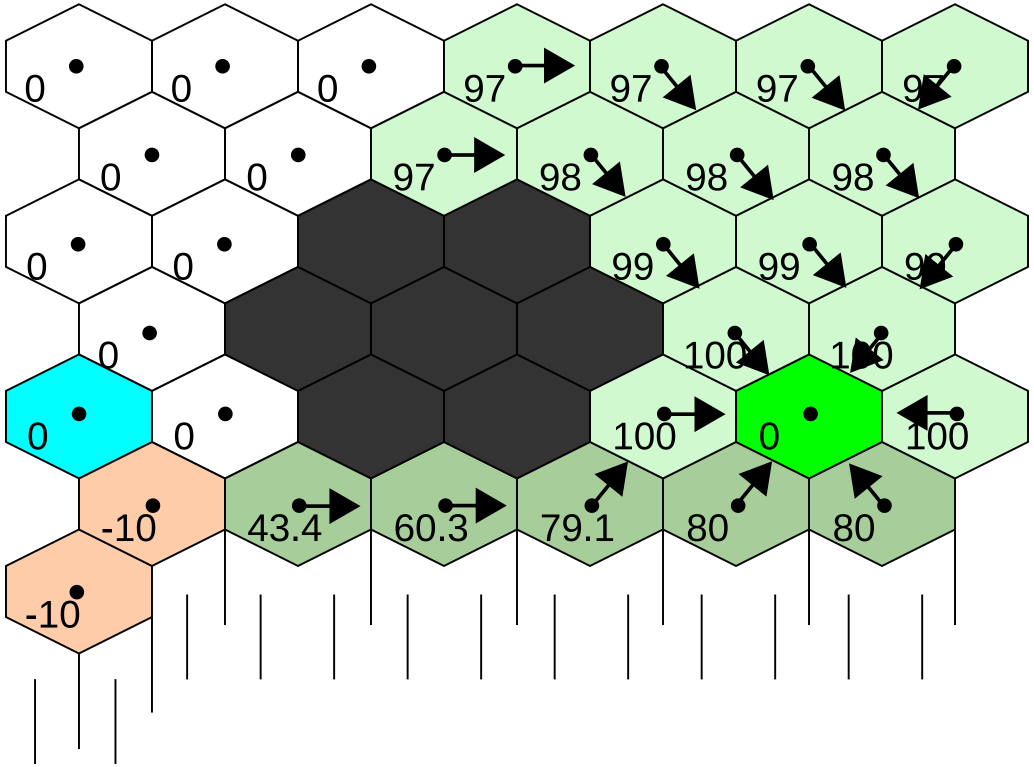
Value-Iteration: first iteration

Example: Dangerous move.



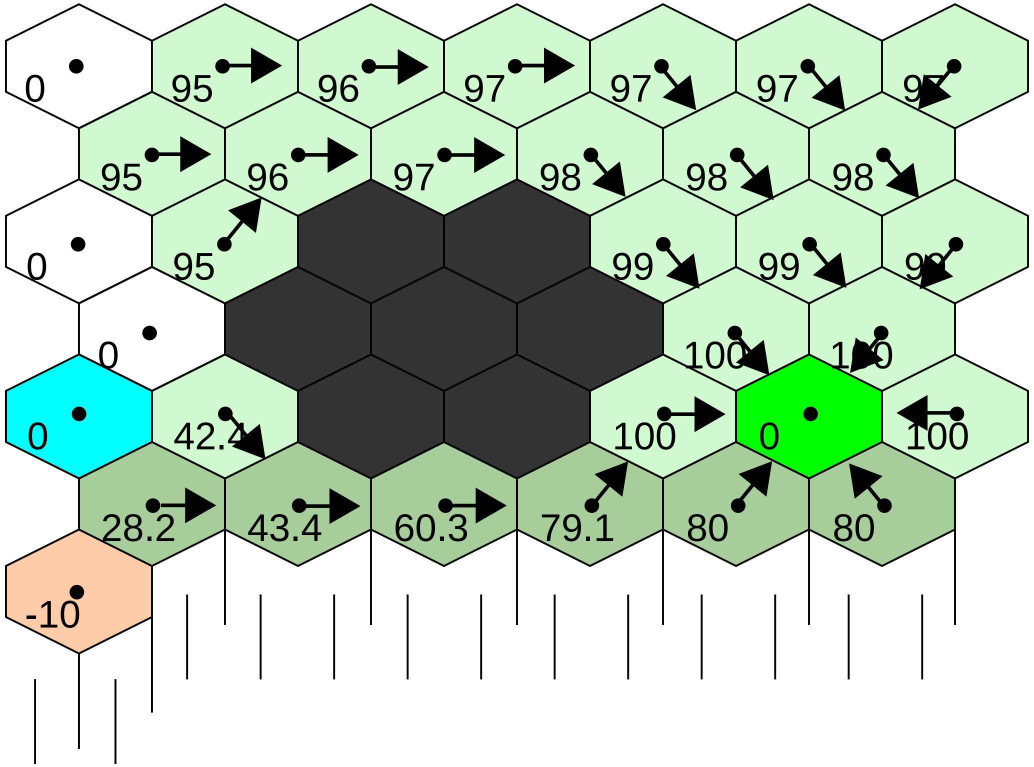
Value-Iteration: second iteration

Example: Dangerous move.



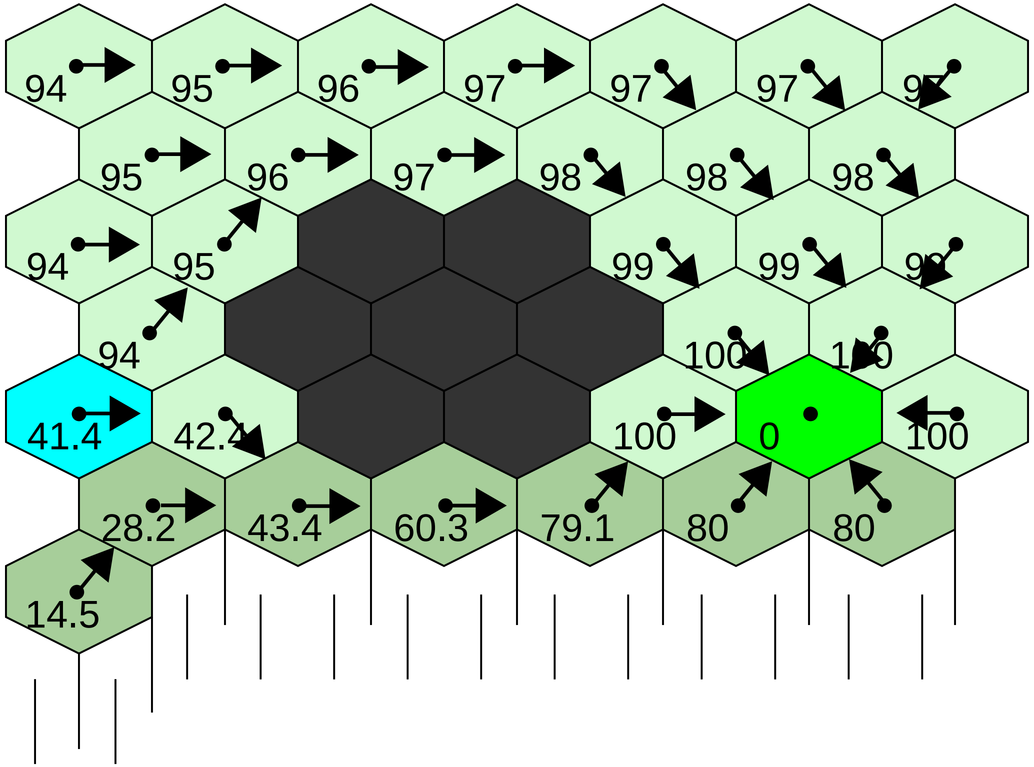
Value-Iteration: 4^{th} iteration

Example: Dangerous move.



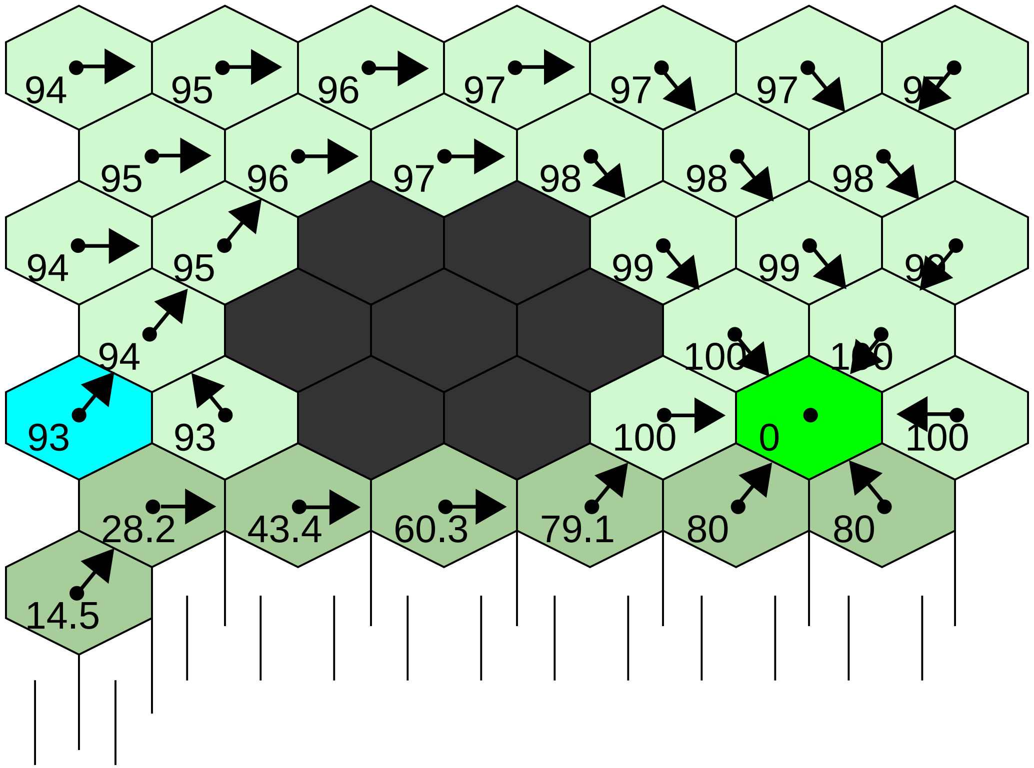
Value-Iteration: 6^{th} iteration

Example: Dangerous move.



Value-Iteration: 7^{th} iteration

Example: Dangerous move.



Value-Iteration: 8^{th} iteration

Solving MDP: Policy Iteration

Input: an **MDP:** $\langle S, A, T, R \rangle$; precision error: ϵ ; discount factor: γ ; initial **V(s)**

1. Compute $\pi(s)$ according to $V(s)$, for each state $s \in S$
2. Repeat until $\pi(s)$ is stable:
   - Update $V(s)$ with $\pi(s)$ at ϵ error, for each state $s \in S$
   - Update $\pi(s)$ according to $V(s)$, for each state $s \in S$

Output: an optimal π^* and associated V-values

Ok now learn the model...

- Define the state-space (small but covering).
- Define the action-space.
- Explore the system:
  - Compute the average rewards $R(s, a)$.
  - Compute all transition probability $T(s, a, s')$

Learn the transition

The transition function is the core object to learn.

It is a 3-dimention structure of floating point values (probabilities).

$$|S|^2 \times |A| \text{ values.}$$

A simble game as **421** with **168** states and **8** actions
would requires **225 792** values.

Luky for us, in application, most of the transitions are null (ie. imposible),
and there are some similarity in the 'value flow'.

let try it on 421 game