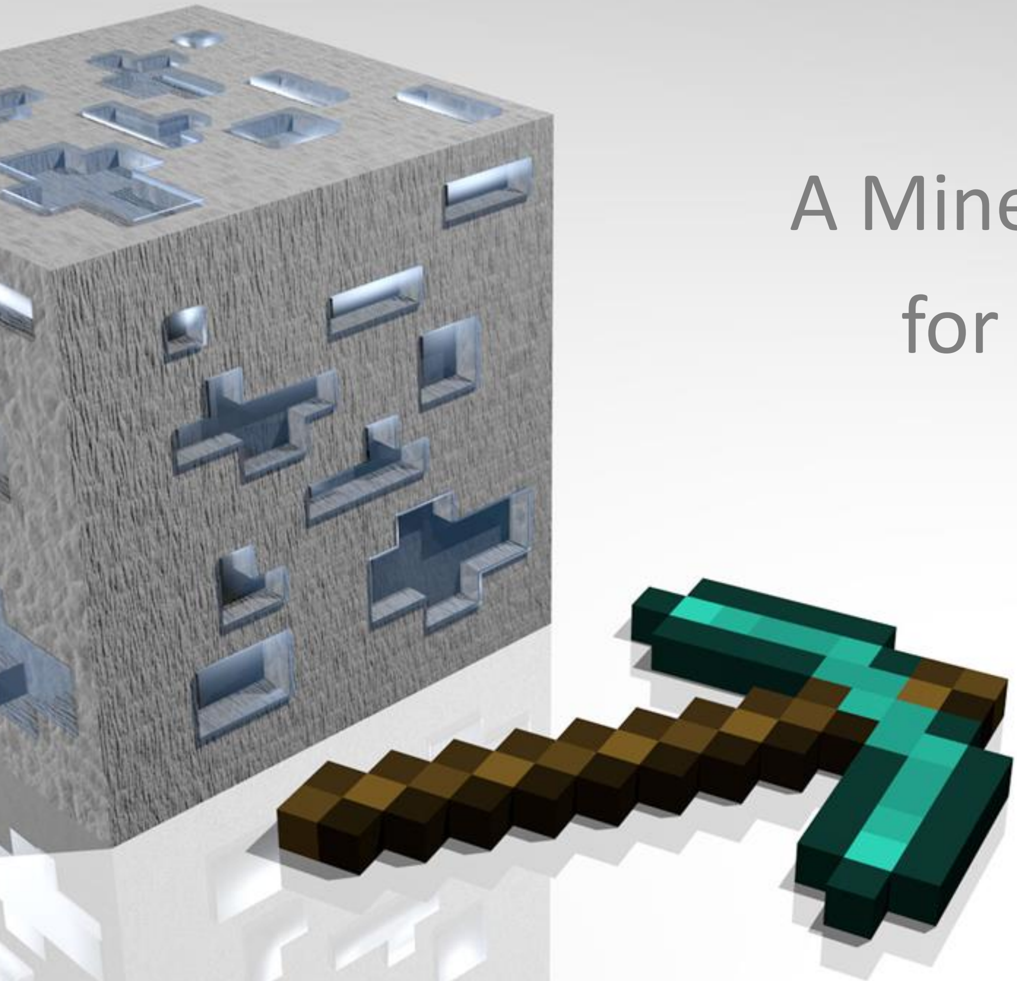
Sequential decision making under uncertainty in randomly generated games

A Minecraft intelligent agent for resource gathering



Université de Liège

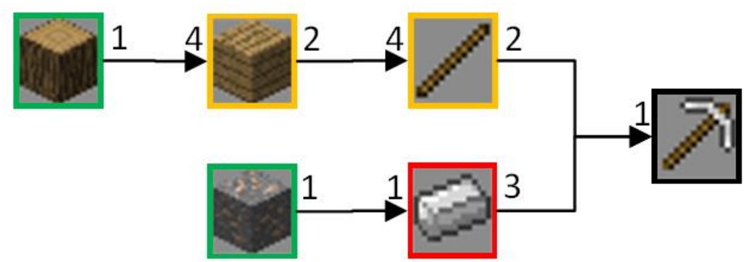
Lorent Pierre

Summary

- Minecraft
- Agent goal
- Modelisation
- RL algorithm choice
- Results
- Conclusion

Minecraft

- 3D video game
- Randomly generated worlds
- Resources can be gathered
- Technology tree to build tools

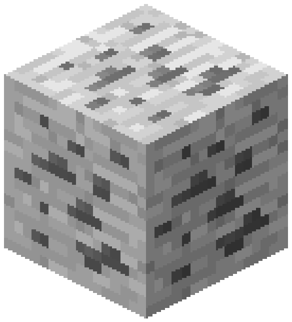


Agent goal

Collect efficiently a given resource :

- Selected resource is coal
  - Common resource
  - Used for many purpose

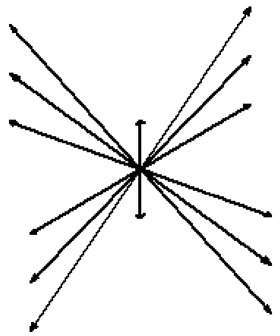
- Efficient means gather as much coal as possible in a given amount of time.



Modelisation

actions

- Move in a specific direction
  - 14 directions available
  - World edition allowed
  - Use of a pathfinding algorithm



- Gather a given resource
  - Dirt, rock, wood, iron, coal
  - Resource must be visible
- Craft tools
  - Pickaxes, axes or shovels
  - Different qualities using different resources
  - These resources must be collected

Modelisation

state-space decomposition

$$\mathbb{S} = \underbrace{\mathbb{R}}_{\text{Real elapsed time}} \times \mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathcal{I} \times \mathcal{E}$$

- **1631.042 ;**
- 3.0 ; -13.0 ; -21.0 ; -3.0 ; 3.0 ; 0.0 ;
- 298.0 ; 112.0 ; 0.0 ; 0.0 ; 4.0 ; 1.0 ; 0.0 ;
- 23.0 ; 67.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 28.0 ; 0.0 ; 0.0 ;
- 45.0 ; 282.0 ; 0.0 ; 5.0 ; 6.0 ;
- 1.0 ; 1.0 ; 1.0 ; 1.0 ;
- 4.04 ; 0.93 ; 4.26 ; 1.16 ; 5.66 ; 1.79 E308 ; 5.77 ; 1.79 E308 ;
  1.34 ; 1.52 ; 5.25 ; 2.52 ; 1.79 E308 ; 2.74 ;
- 2.27 ; 5.12 ; 4.33 ; 2.88 ; 2.49 ; 9.71 ; 8.40 ; 3.85 ;
  3.77 ; 9.97 ; 9.71 ; 3.76 ; 5.02 ; 8.21

Modelisation

state-space decomposition

$$\mathbb{S} = \mathbb{R} \times \underbrace{\mathbb{Z}^3 \times \mathbb{Z}^3}_{\text{Position and delta from previous action}} \times \mathcal{I} \times \mathcal{E}$$

- 1631.042 ;
- **3.0 ; -13.0 ; -21.0 ; -3.0 ; 3.0 ; 0.0 ;**
- 298.0 ; 112.0 ; 0.0 ; 0.0 ; 4.0 ; 1.0 ; 0.0 ;
- 23.0 ; 67.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 28.0 ; 0.0 ; 0.0 ;
- 45.0 ; 282.0 ; 0.0 ; 5.0 ; 6.0 ;
- 1.0 ; 1.0 ; 1.0 ; 1.0 ;
- 4.04 ; 0.93 ; 4.26 ; 1.16 ; 5.66 ; 1.79 E308 ; 5.77 ; 1.79 E308 ;
  1.34 ; 1.52 ; 5.25 ; 2.52 ; 1.79 E308 ; 2.74 ;
- 2.27 ; 5.12 ; 4.33 ; 2.88 ; 2.49 ; 9.71 ; 8.40 ; 3.85 ;
  3.77 ; 9.97 ; 9.71 ; 3.76 ; 5.02 ; 8.21

Modelisation

state-space decomposition

$$\mathbb{S} = \mathbb{R} \times \mathbb{Z}^3 \times \mathbb{Z}^3 \times \underbrace{\mathcal{I}}_{\text{Inventory descriptor}} \times \mathcal{E}$$

- Amount of resources
- Tools durability

- 1631.042 ;
- 3.0 ; -13.0 ; -21.0 ; -3.0 ; 3.0 ; 0.0 ;
- **298.0 ; 112.0 ; 0.0 ; 0.0 ; 24.0 ; 1.0 ; 0.0 ;**
- **23.0 ; 67.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 28.0 ; 0.0 ; 0.0 ;**
- 45.0 ; 282.0 ; 0.0 ; 5.0 ; 6.0 ;
- 1.0 ; 1.0 ; 1.0 ; 1.0 ;
- 4.04 ; 0.93 ; 4.26 ; 1.16 ; 5.66 ; 1.79 E308 ; 5.77 ; 1.79 E308 ;
  1.34 ; 1.52 ; 5.25 ; 2.52 ; 1.79 E308 ; 2.74 ;
- 2.27 ; 5.12 ; 4.33 ; 2.88 ; 2.49 ; 9.71 ; 8.40 ; 3.85 ;
  3.77 ; 9.97 ; 9.71 ; 3.76 ; 5.02 ; 8.21

Modelisation

state-space decomposition

$$\mathbb{S} = \mathbb{R} \times \mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathcal{J} \times \underbrace{\mathcal{E}}_{\text{Environment descriptor}}$$

- Amount of resource block in FOV
- Type of environment
- Displacement cost
- Mean view distance

- 1631.042 ;
- 3.0 ; -13.0 ; -21.0 ; -3.0 ; 3.0 ; 0.0 ;
- 298.0 ; 112.0 ; 0.0 ; 0.0 ; 4.0 ; 1.0 ; 0.0 ;
- 23.0 ; 67.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 28.0 ; 0.0 ; 0.0 ;
- **45.0 ; 282.0 ; 0.0 ; 5.0 ; 6.0 ;**
- **1.0 ; 1.0 ; 1.0 ; 1.0 ;**
- **4.04 ; 0.93 ; 4.26 ; 1.16 ; 5.66 ; 1.79 E308 ; 5.77 ; 1.79 E308 ; 1.34 ; 1.52 ; 5.25 ; 2.52 ; 1.79 E308 ; 2.74 ;**
- **2.27 ; 5.12 ; 4.33 ; 2.88 ; 2.49 ; 9.71 ; 8.40 ; 3.85 ; 3.77 ; 9.97 ; 9.71 ; 3.76 ; 5.02 ; 8.21**

Modelisation

reward

- The agent must be rewarded when gathering the target resource
- « Time is money ».

$$r(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = \beta * \Delta i_{goal} - (1 - \beta) * \Delta T$$

RL algorithm choice

Main challenges :

- High-dimensional state-space
- Complex sequence of actions to be efficient

  → Gather enough wood when the entity is above the ground
  → Build a wood pickaxe
  → Dig into the ground to reach stone
  → Gather some stone
  → Build a stone pickaxe

➡ Use of expert trajectories as bootstrap samples

RL algorithm choice

Fitted-Q iteration :

- Build an approximation of long-term expected reward when taking action a_t from state s_t :

$$Q^i(\mathbf{s}_t, a_t) = r_t + \gamma \max_{a \in \mathcal{A}} Q^{i-1}(\mathbf{s}_{t+1}, a)$$

- Select most valuable action thanks to :

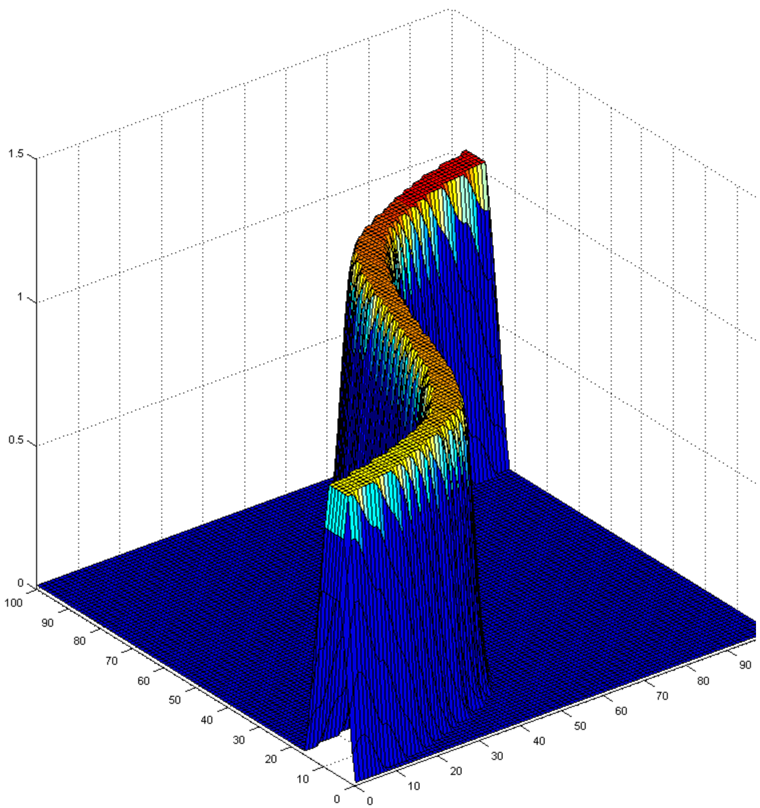
$$a_{ideal}(\mathbf{s}_t) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(\mathbf{s}_t, a)$$

- Use a regression algorithm to estimate Q from the provided samples
  - Extra-trees for example

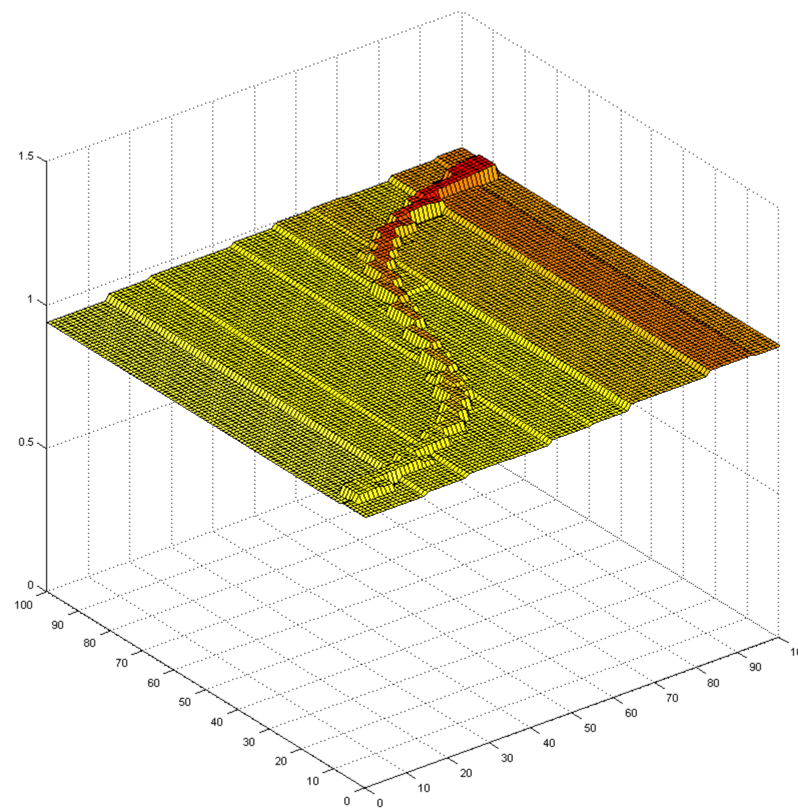
Existing examples of successful applications on high-dimensional problems

RL algorithm choice

Desired behaviour



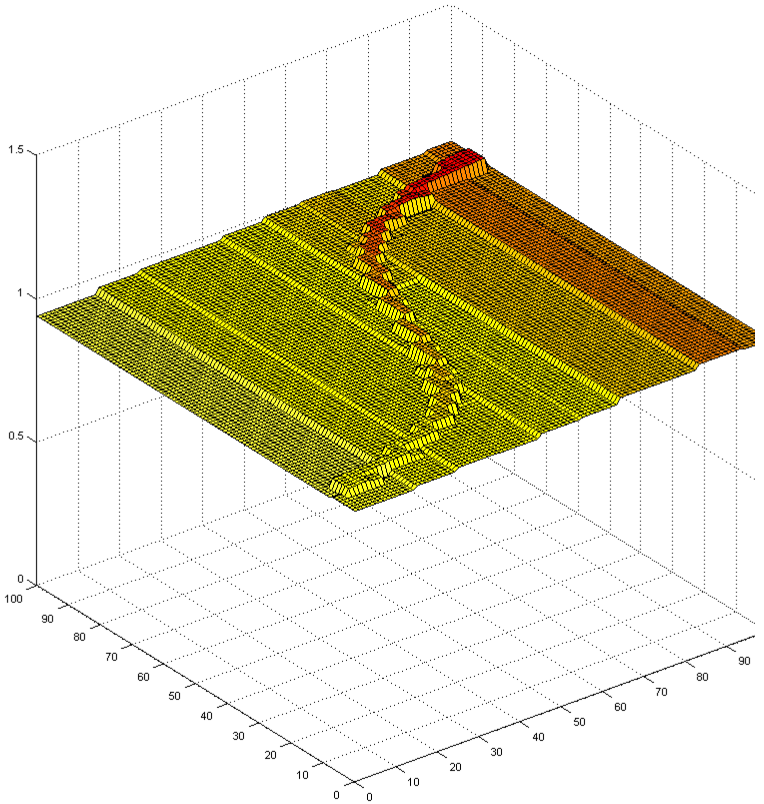
Q-function as seen from the player perspective



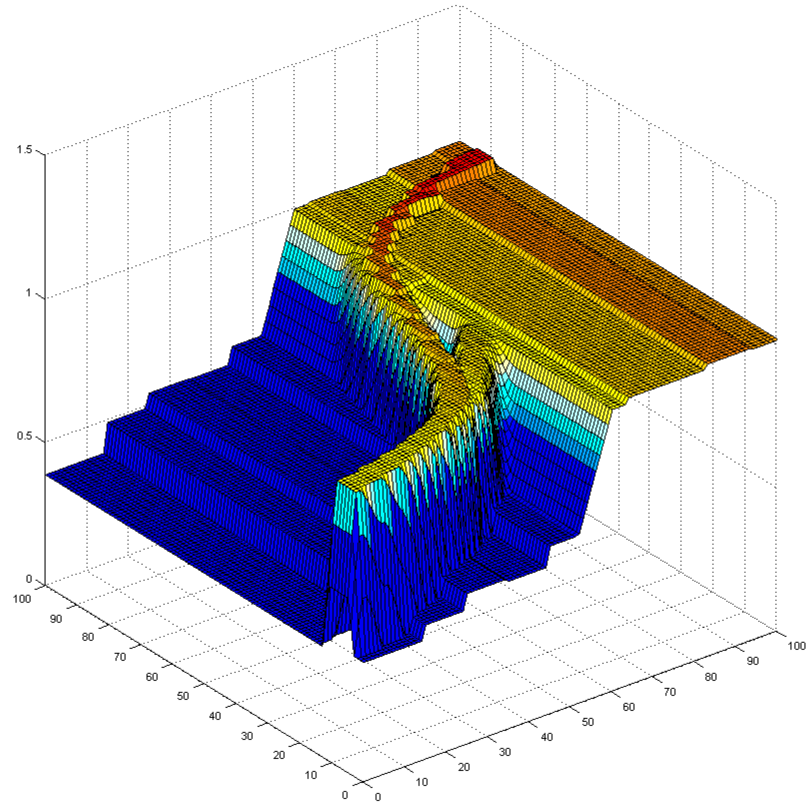
Q-function learned with expert trajectories only

RL algorithm choice

Desired behaviour



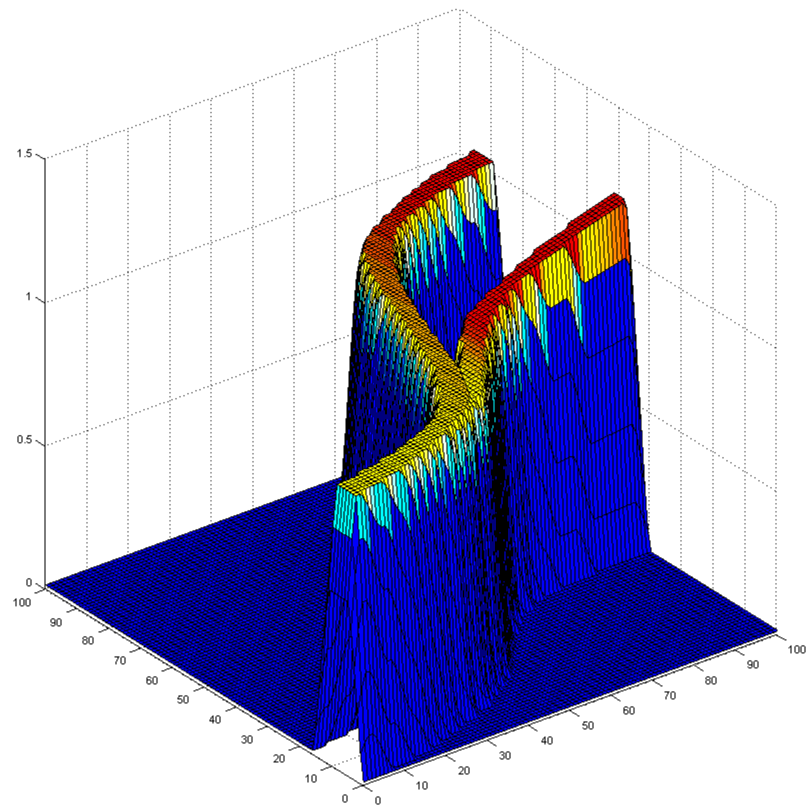
Q-function learned with expert trajectories only



Q-function during the learning phase

RL algorithm choice

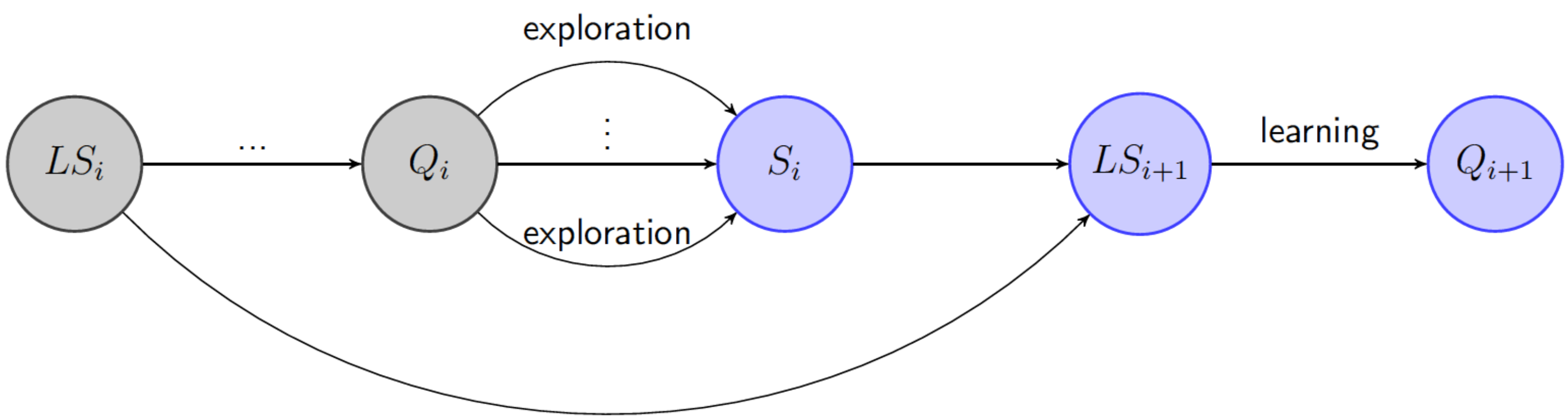
Desired behaviour



Resulting Q-function

RL algorithm choice

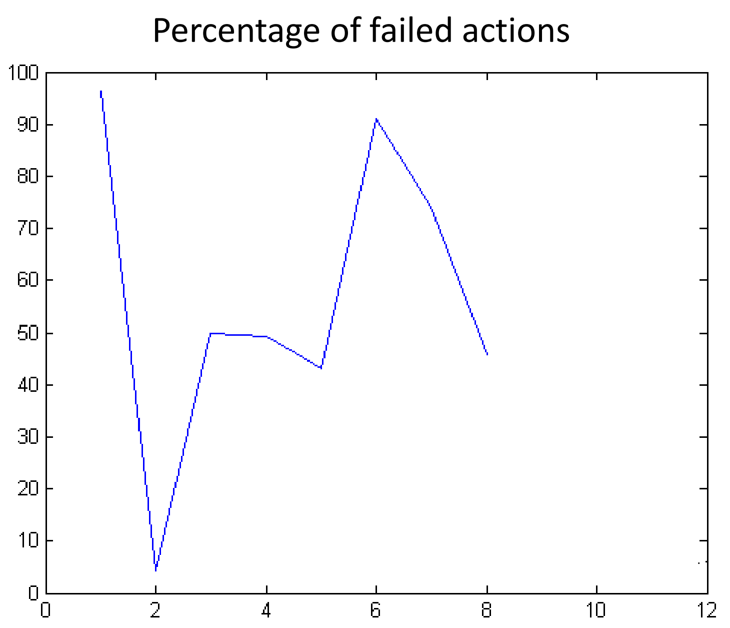
- Learning process



Results

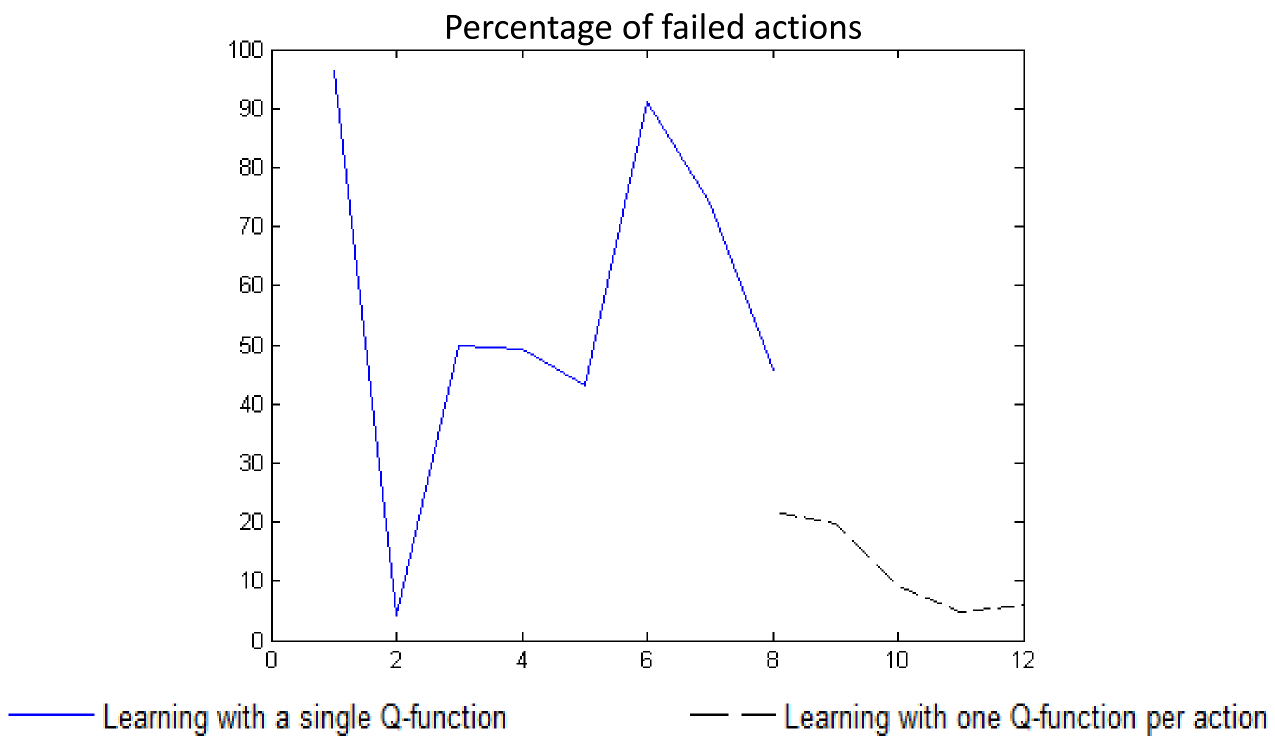
- Evolution of the agent behaviour :
  - Trying to collect coal in any situation
  - Digging without tools
  - Collecting wood without using it
  - Trying to build tools without any resources

- Still difficult for the agent to understand the purpose of each action, resulting in high failure rate and poor accuracy in decisions.

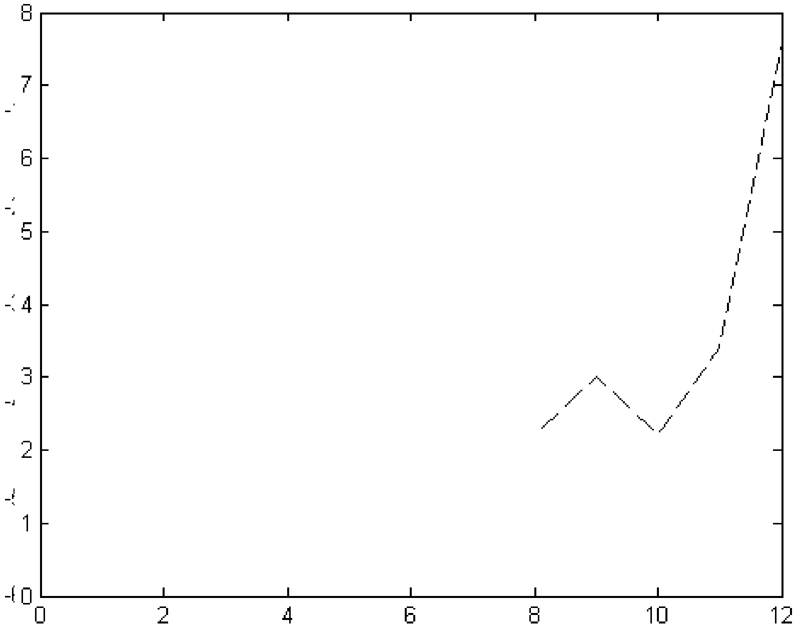


Results

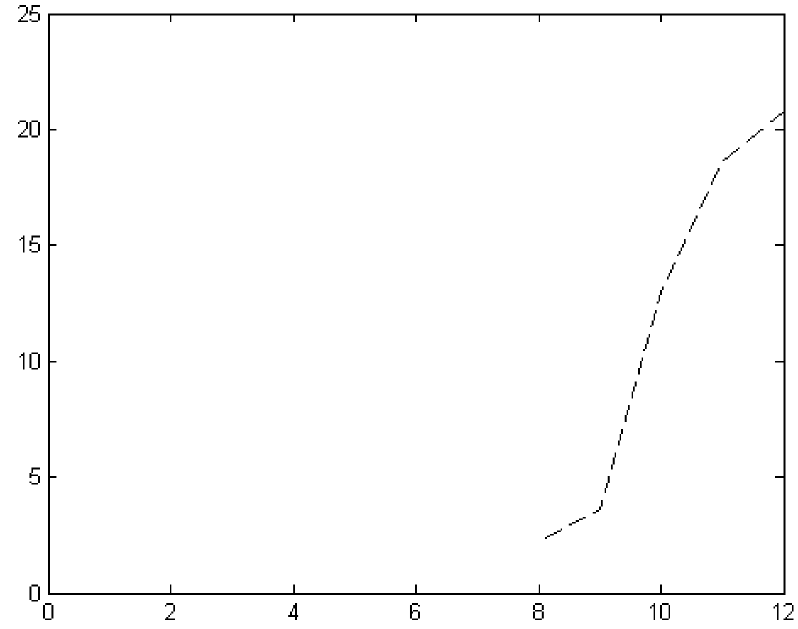
- Solution : fit a different Q-function per action.
  - Allow the resulting policy to be much more precise when predicting the effect of an action without the influence of the others.



Results



Coal gathered in100 actions



Coal gathered in20 minutes

Learning with a single Q-function

Learning with one Q-function per action

Conclusion

- The agent learned the system dynamic despite the dimension of the problem

- Some situations may be improved

  → Coal blocks are sometimes missed due to « blind » displacements and the stationnarity of the decision policy.

Conclusion

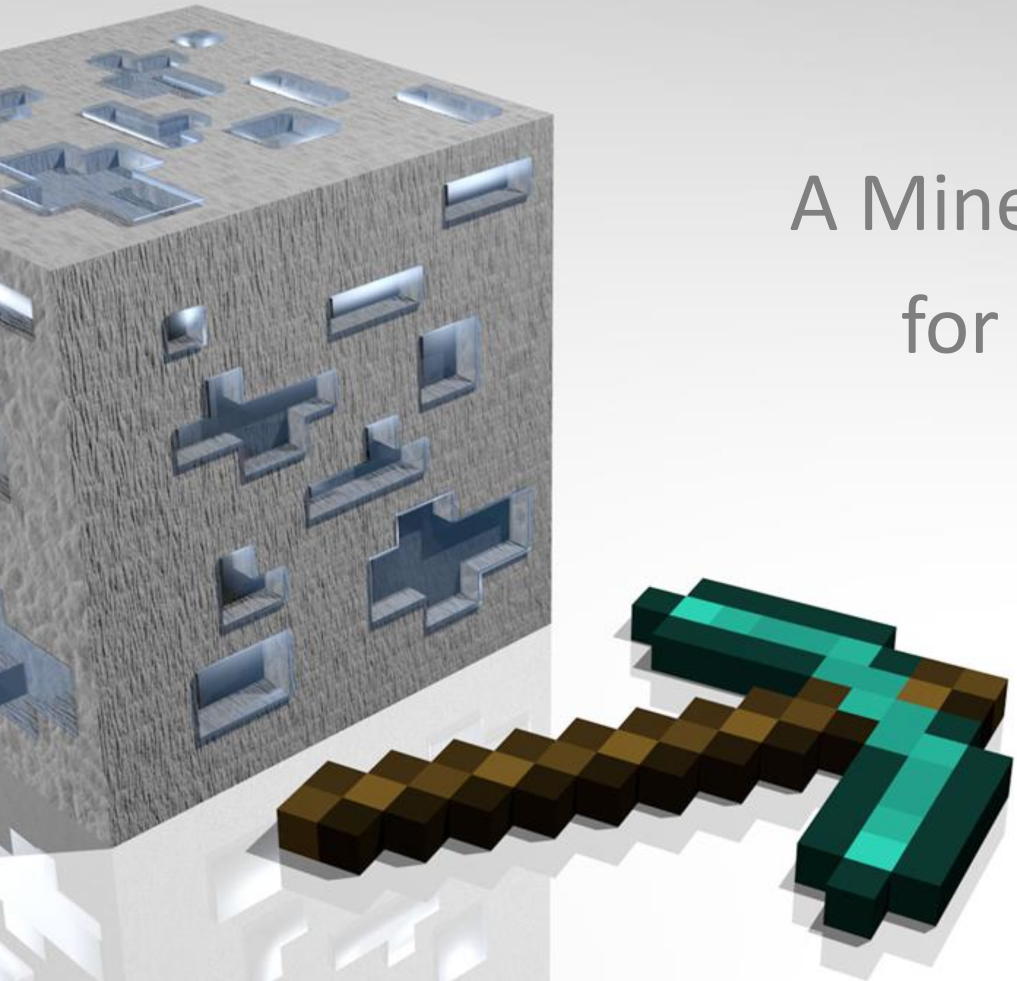
Future work

- Restart the learning process using per-action Q-functions from step 0.
- Restart the learning process without the expert samples

Thank you for your attention

Illustration credits:
http://fc04.deviantart.net/fs71/f/2012/014/8/b/minecraft_diamond_ore_wallpaper_by_swatspeedman-d4m0un0.png

Sequential decision making under uncertainty in randomly generated games



A Minecraft intelligent agent for resource gathering

Université de Liège

Lorent Pierre