MovitIt

An HackaGames game.

Guillaume Lozenguez

@imt-nord-europe.fr

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

1. **Applying Q-Learning**
2. **Model-Based Decision Making**

Basic State Representation

- Robot position (6×4)
- Robot goal (6×4)
- Robot direction (6)
- Obstacles Positions (6×4) *(6 obstacles)*
- Humans' position (6×4), direction (6) *(2 humans)*

States: $24^{(2+6+2)} \times 6^3 = 1.3 \times 10^{16}$

Relative State Representation

- Robot goal direction (6), distance (16)
- Distance-1 Cells: obstacle? (2^6)
- Humans' position-diretion (6), position-distance (16), movement-direction (6)

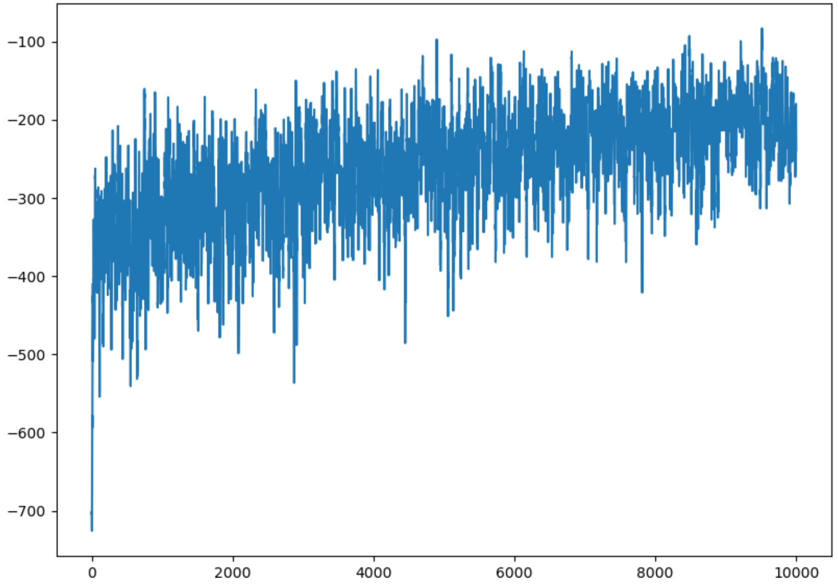
States: $6^5 \times 16^3 \times 2^6 = 2.0 \times 10^9$

A huge gain on the number of state $+$ promising factorization.

However: not covering -> no guarantee

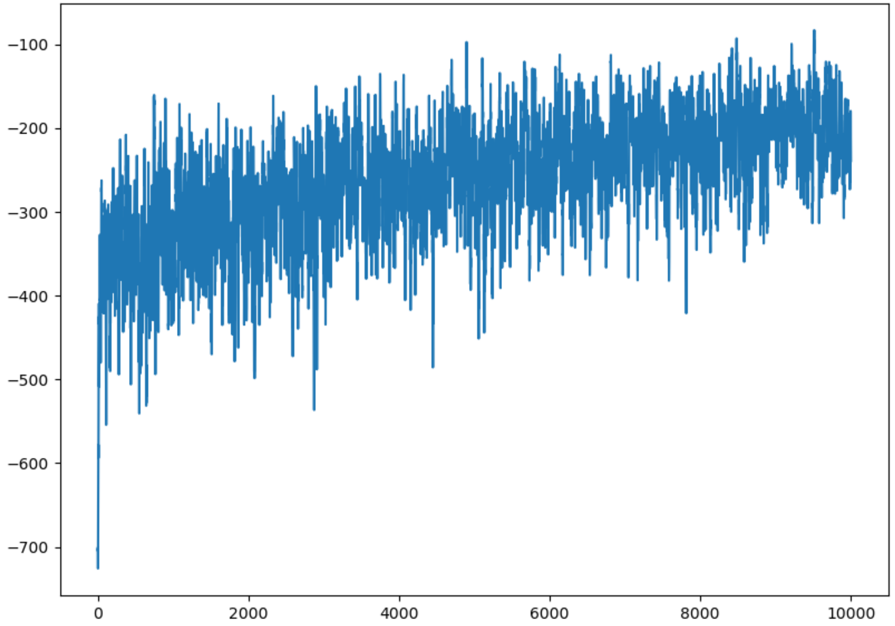
QLearning based on Relative State Representation

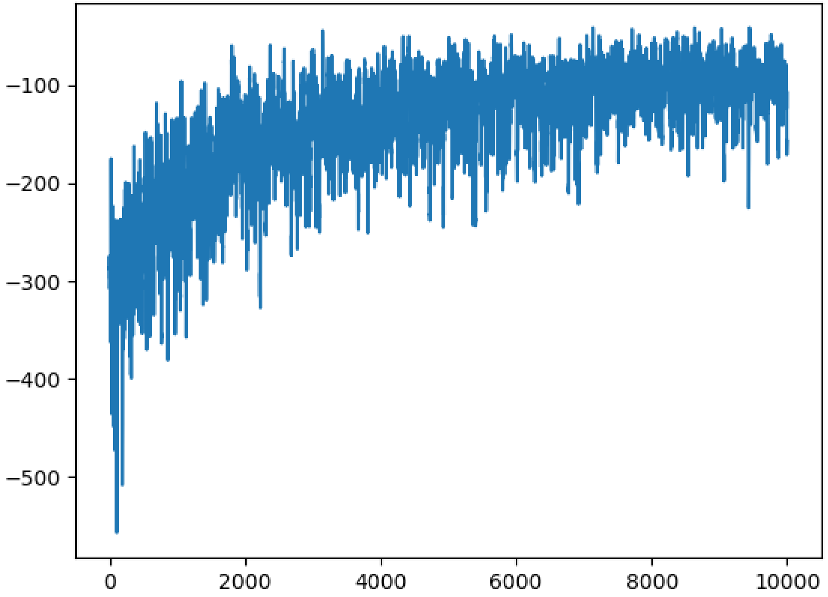
Average score over 10 000 games of 10 cycles.



QLearning based on Relative State Representation

Qlerning Relative-basic versus Relative-limited





1. **Applying Q-Learning**
2. **Model-Based Decision Making**

Simulating Action MoveIt ?

Is it possible to model and simulate GameMoveIt ?

Simulating Action MoveIt

Board builtin function:

```
   collisions= self.board().multiMoveHumans( humanMoves )
   collisions+= self.board().multiMoveRobots( robotMoves )
```

- *moves* list of start position and direction.

$$moves = [[x1, y1, dir1], [x2, y2, dir2] \ldots]$$

Simulation squeletom:

```
def simulate(board, robotMoves):
   copiedBoard= copy(board)
   humanMoves= generateHumanMove(board)
   collisions= board.multiMoves...
   return collisions, copiedBoard
```