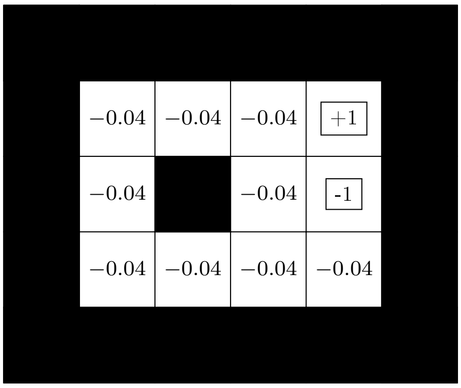
Artificial Intelligence

Markov Decision Processes lab

The objective is to implement and evaluate the value and policy iteration algorithms for Markov Decision Processes.

We address a small example from the book *Artificial Intelligence – A Modern Approach* in an as generic as possible approach.

We model the environment as a tiled rectangular area of length L and height H. You may want to reuse and adapt the data structures and functions from the *Path-finding lab*. We focus, in particular, on the following small 6×5 environment :



The number in each tile represents the immediate reward obtained when moving to it. The black tiles are impassable walls. Note that, as in the *Path-finding lab*, we assume the environment to be surrounded by walls. The tiles with rewards $+1$ and -1 are terminal nodes : when the robot reaches them it can never move again. Hence, the utility for these tiles (from iteration 1 on) is just the immediate reward.

We assume that each time the robot tries to move in one direction, there is a 10% chance that it goes left (relatively to the direction chosen) instead of straight ahead and 10% chance it goes right (still relative). If this makes it go into a wall, it just stays put (does not move).

Q1. Implement value iteration for the corresponding MDP. Display the computed policy at each iteration using e.g. characters 'v', '<', '>', '^'. Display also the number of iterations required to converge. Use values $\gamma = 0.99$ and $\epsilon = 0.01$.

Ath the end of the computation, also display the computed utilities.

Q2. Experiment with other rewards. Try in particular to vary the reward for "regular" tiles. What happens when it is positive ? And when it is much less than -0.04 (say -2) ? Can you find more intermediate situations ?

Q3. Implement policy iteration for the corresponding MDP. Use a simplified version of the function written in Q1. to compute the utilities of policies up to ϵ. Display the computed policy at each iteration and the number of iterations required to converge. Compare with value iteration.

Q4. Implement the Q-learning reinforcement learning algorithm, using an ϵ-greedy exploration policy (recall that ϵ has not the same role as in the previous questions here) ;

Q5. Assume we start in the bottom right-most tile and compute the best strategy according to the algorithm of Q4. Use $\epsilon = 0.1, \alpha = 0.05, \gamma = 0.99$ and 10000 runs.