# Project IV - Reinforcement Learning

Carlos Matheus Barros da Silva[1]

Igor Bragaia[1]

Prof. Paulo Andre Castro

## I. OBJECTIVE

This works aims to elucidate the Reinforcement Learning approach to solve a Markov Decision Process in a stochastic grid problem. In order to do that, the problem has been modeled and simulated validating the correct solution convergence. Afterall, an optimal policy to this problem has been proposed.

## II. PROBLEM

On the proposed problem, there is a grid as follows containing cells with different properties.

[1]Computer Engineering Bachelor Student of ITA

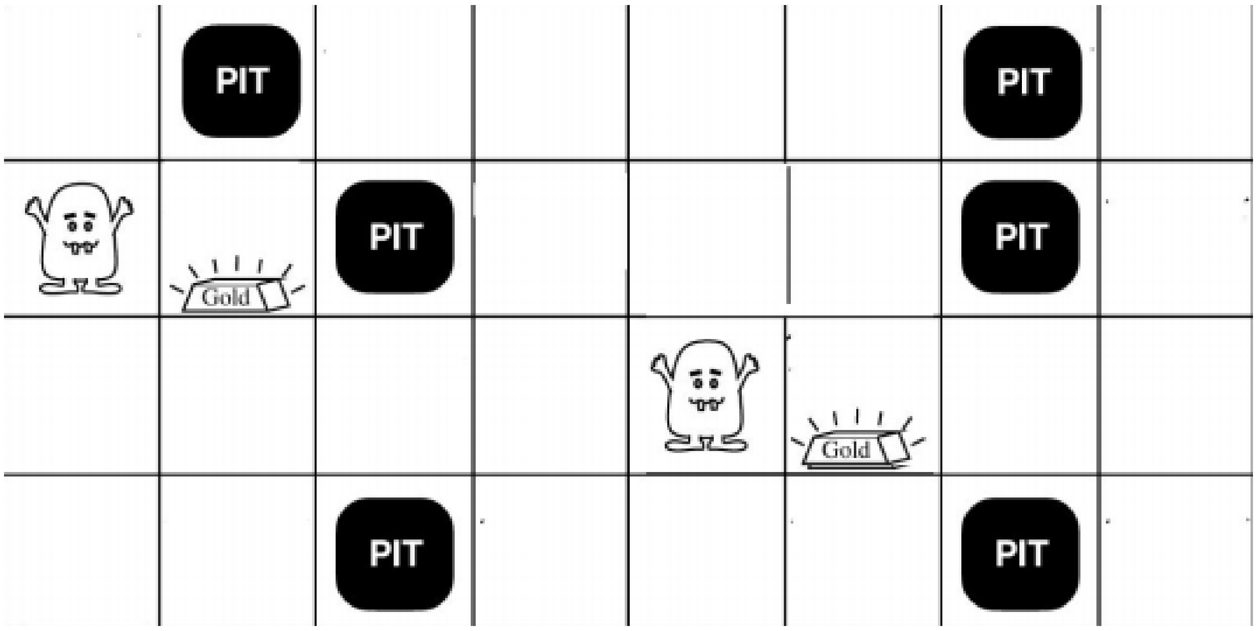


Fig. 1. Proposed problem

Each cell represents a possible position for a robot. When navigating through this grid, each movement has a cost of $-0.1$ and when the robot reach out a PIT cell it gets a reward of $-50$, when it reach a Monster cell it gets a reward of $-100$ and when it reach out a Gold cell it gets a reward of $+100$. Also, it is a stochastic grid then when a movement is about to be done, it has $0.7$ probability of in fact doing this movement, $0.2$ probability of sliding to its left cell and $0.1$ probability of sliding to its right cell. Finally, if the robot tries to move out of the board, it has an extra reward of $-0.1$.

On this project, we will propose the optimal grid navigation policy in order to accumulate a maximum reward.

## III. Proposed model

This problem has been modeled as a first order Markov Decision Process in which each board state only depends on the previous board state. In order to do that, at each iteration of the reinforcement learning, we compute the next board state looping over the board following the rule that the next state utility for a specific cell is the maximum value among the stochastic utility for an up, down, left or right movements.

A stochastic utility for a grid position and its movement is defined as the reward for the position plus the learning rate multiplied by the sum of the following values: $-1$ if the movement is out of the board (then it remains the at the same position), $0.7$ multiplied by the current utility for the position of the desired movement, $0.2$ multiplied by the current utility for the position on the left side of the current position and $0.1$ multiplied by the current utility for the position on the right side of the current position. Furthermore, when the robot moves to a position that contains Monster, PIT or Gold, the game should be restarted and the robot should go to a random cell chosen uniformly. In order to turn this into a quantitative meaning, on this case the learning rate will be multiplied only by the average of the current utility of all board cells. Also, if the desired movement is out of the board, the robot is supposed to remain on the same position.

Then, for each reinforcement learning iteration, the algorithm calculates the utilities sum for the current board state subtracted by the utilities sum for the previous board state. The convergence is found when this metric is zero, what means that the board state will not change anymore.

Finally, when the reinforcement learning converges, an optimal traversal policy can be calculated. On the optimal policy, for each cell the robot should move always to its neighbor with higher utility value.

## IV. Results

Choosing a learning rate of $1.0$, the reinforcement learning metric convergence has not decreased to zero as expected but it has converged to zero correctly using a learning rate of $0.8$.

Thus, for a learning rate of $0.8$, reinforcement learning metric convergence has converged as follows over iterations.

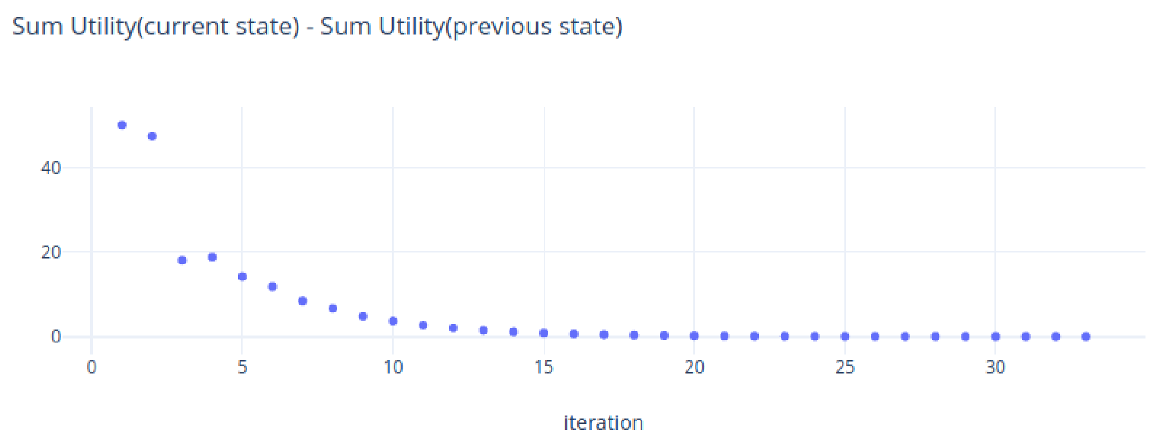
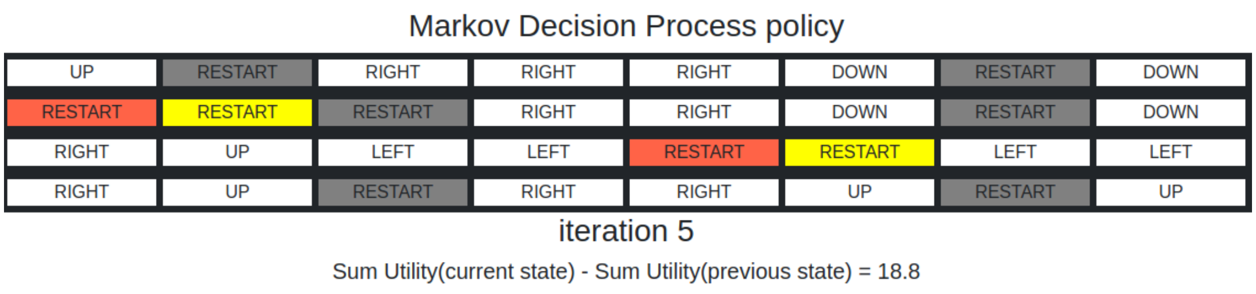


Fig. 2. Reinforcement learning metric convergence

As an example, there is a non-optimal policy snapshot on the 5th iteration. At this representation, a red cell contains Monster, a red cell contains Gold and a grey cell contains PIT.


Markov Decision Process policy

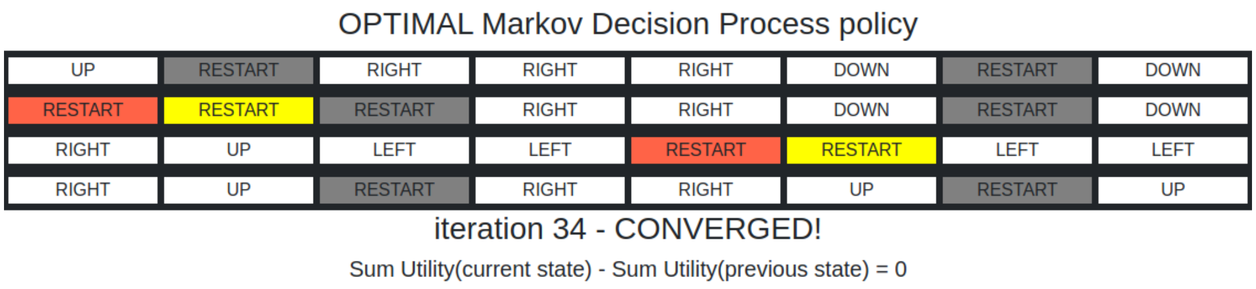
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| UP | RESTART | RIGHT | RIGHT | RIGHT | DOWN | RESTART | DOWN |
| RESTART | RESTART | RESTART | RIGHT | RIGHT | DOWN | RESTART | DOWN |
| RIGHT | UP | LEFT | LEFT | RESTART | RESTART | LEFT | LEFT |
| RIGHT | UP | RESTART | RIGHT | RIGHT | UP | RESTART | UP |

iteration 5

Sum Utility(current state) - Sum Utility(previous state) = 18.8

Fig. 3. Policy on the 5h iteration

Finally, there is the optimal policy snapshot on the 34th iteration after reinforcement learning convergence.


OPTIMAL Markov Decision Process policy

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| UP | RESTART | RIGHT | RIGHT | RIGHT | DOWN | RESTART | DOWN |
| RESTART | RESTART | RESTART | RIGHT | RIGHT | DOWN | RESTART | DOWN |
| RIGHT | UP | LEFT | LEFT | RESTART | RESTART | LEFT | LEFT |
| RIGHT | UP | RESTART | RIGHT | RIGHT | UP | RESTART | UP |

iteration 34 - CONVERGED!

Sum Utility(current state) - Sum Utility(previous state) = 0

Fig. 4. Optimal policy on the 34th iteration

## V. Conclusions

The work was quite enlightening when it comes to Reinforcement Learning comprehension and exemplification. Its difficulty was right-minded, although the amount of code required made the work duller than complex.

## VI. Implementation Description

The entire implementation was made using JavaScript. Due to this choice, it was easier to create a visualizable simulation to understand the algorithm progress. The complete code can be found at the Github project the Github project https://tinyurl.com/y5d6yxza.