**Laboratório 11 – Programação Dinâmica**

**Inteligência Artificial para Robótica Móvel – CT-213**

**Aluno: Caio Graça Gomes**

**Professor: Marcos Ricardo Omena de Albuquerque Maximo**

## Introdução:

Nesse laboratório, teve-se por objetivo implementar três algoritmos de programação dinâmica para solução de um Processo Decisório de Markov (*MDP*). Os algoritmos implementados serão os de avaliação de política (*policy evaluation*), iteração de valor (*value iteration*) e iteração de política (*policy iteration*). Deve-se avaliar e encontrar políticas ótimas em uma ambiente de *grid world*.

## Metodologia e descrição em alto nível do algoritmo utilizado:

O problema consistiu em um *grid world* com 5 ações possíveis: STOP, UP, LEFT, RIGHT e DOWN, além de obstáculos e limites do *grid* como barreiras. A ação STOP é sempre executada com probabilidade 1, enquanto as outras ações têm uma certa probabilidade de serem executadas corretamente. Se uma ação não é executada corretamente, o resultado das demais ações acontece com igual probabilidade. O MDP tem fator de desconto $\gamma$ (que será adotado como 1 em uma parte do experimento, e 0,98 como outra parte) e a recompensa é -1 para cada instante que o agente passa em uma célula que não é a objetivo. Há uma única célula objetivo no *grid*, onde o agente recebe recompensa 0.

O algoritmo *policy evaluation* se baseia na equação de Bellman de expectativa, será adotada uma solução iterativa (Gauss-Jacobi) do sistema linear associado às equações de Bellman.

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a) + \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_\pi(s')$$

O algoritmo de iteração de valor se baseia em iterar diretamente de acordo com a equação da otimalidade de Bellman:

$$v_*(s) = max_{a \in A}(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s'))$$

Já o algoritmo de iteração de política alterna entre avaliação de política (3 vezes, nesse laboratório) e aprimoramento de política. Uma política gulosa *greedy* é obtida ao fazer:

$$\pi'(s) = argmax_{a \in A}(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s'))$$

Quando existir mais de uma ação ótima, é dada a mesma probabilidade para cada uma ser executada.

Os três algoritmos foram testados em *test_dynamic_programming.py*, sob duas diferentes circunstâncias: primeiramente fez-se CORRECT_ACTION_PROBABILITY = 1.00 e GAMMA = 1.00, e então CORRECT_ACTION_PROBABILITY = 0.80 e GAMMA =0.98. Ao término, foram comparados seus desempenhos.

## Resultados do test_dynamic_programming.py:

Figura 1: *Policy Evaluation* na primeira condição.

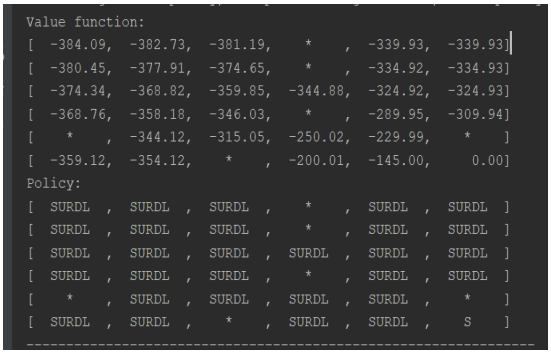



Figura 2: *Value Iteration* na primeira condição.

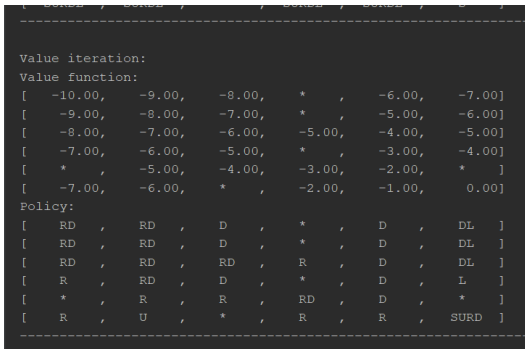



Figura 3: *Policy Iteration* na primeira condição.

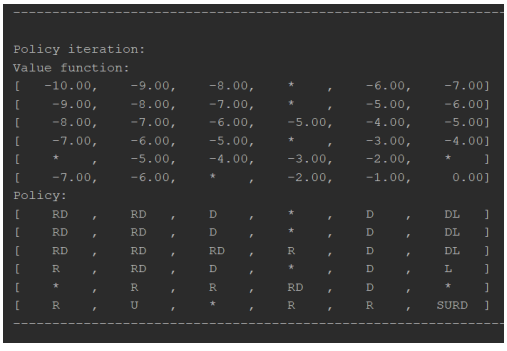



Figura 4: *Policy Evaluation* na segunda condição.

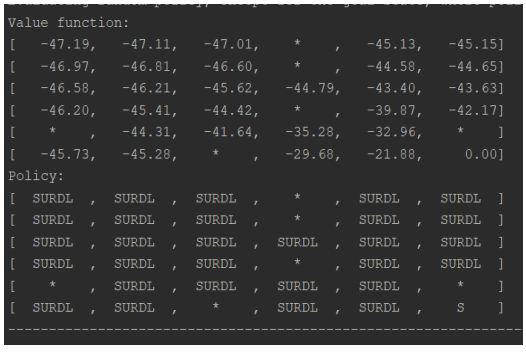
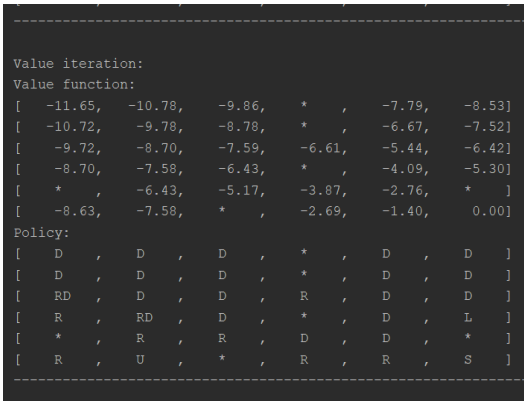
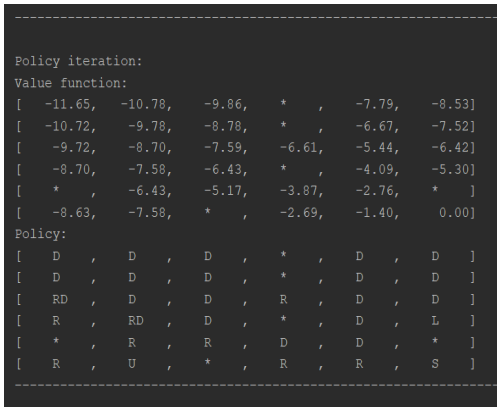



Figura 5: *Value Iteration* na segunda condição.

Figura 6: *Policy Iteration* na segunda condição.




Conforme esperado, os algoritmos de *policy iteration* e *value iteration* convergiram para um mesmo valor, no caso, encontraram os valores e políticas ótimos. Percebe-se ainda, um desempenho não muito razoável do algoritmo de *policy evaluation*, que melhorou seu desempenho (aproximou-se dos valores do *policy* e *value iteration*) ao fazer o gamma ser 0.98.