1 Introdução

O objetivo desse laboratório foi implementar (em *Python*) algorítmos de aprendizado por reforço livre de modelo, isto é, algorítmos que conseguem fazer predições sem conhecer o modelo (diferente do laboratório anterior). Para isso, foram escolhidos dois métodos: *Sarsa* e *Q-learning*, que serão descritos nas próximas seções.

2 Ação gulosa e ε-gulosa

2.1 Descrição

Antes de implementar cada um dos métodos, era necessário implementar métodos que achassem a ação gulosa e ε-gulosa. A ação gulosa é a ação com maior ação valor, enquanto a ação ε-gulosa tem uma probabilidade de $1-\varepsilon$ de realizar a ação gulosa, em que $\varepsilon \in [0,1]$ é um número gerado aleatoriamente.

2.2 Código

```
def epsilon_greedy_action(q, state, epsilon):
    num_actions=q.shape[1]
    p=np.random.random()
    if p<epsilon:
        return np.random.randint(num_actions)
    else:
        return greedy_action(q,state)

def greedy_action(q, state):
    return np.argmax(q[state])
```

3 Sarsa

3.1 Código

```
class Sarsa(RLAlgorithm):
    def __init__(self, num_states, num_actions, epsilon, alpha, gamma):
        super().__init__(num_states, num_actions, epsilon, alpha, gamma)

    def get_greedy_action(self, state):
        return epsilon_greedy_action(self.q, state, self.epsilon)

    def learn(self, state, action, reward, next_state, next_action):
        self.q[state][action]+=self.alpha*(reward
                    +self.gamma*self.q[next_state][next_action]-self.q[state][action])
```

3.2 Resultados e discussão

Após rodar o arquivo de teste a saída foi a seguinte:

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.9691181  -1.99       -3.90924412]
 [-3.70140268 -2.9701     -4.38123866]
 [-4.395632   -3.94039891 -4.35373542]
 [-5.18185387 -4.89208783 -4.89246534]
 [-4.38772189 -4.67715347 -3.94039899]]
```

```
 [-3.50521681 -4.40810661 -2.9701    ]
 [-2.96744345 -3.93810673 -1.99      ]
 [-1.99       -2.9701     -1.        ]
 [ 0.         -0.99       -0.99      ]]
Greedy policy learnt:
[L, L, L, L, L, R, R, R, R, S]
```

Como era de se esperar, o algorítimo conseguiu encontrar uma política para chegar ao destino final, isto é, a última célula à direita.

No teste do carrinho, após 500 iterações (valor padronizado para este laboratório) os resultados estão mostrados nas figuras 1, 2, 3 e 4. Observe que o carrinho consegui executar o caminho corretamente na pista.

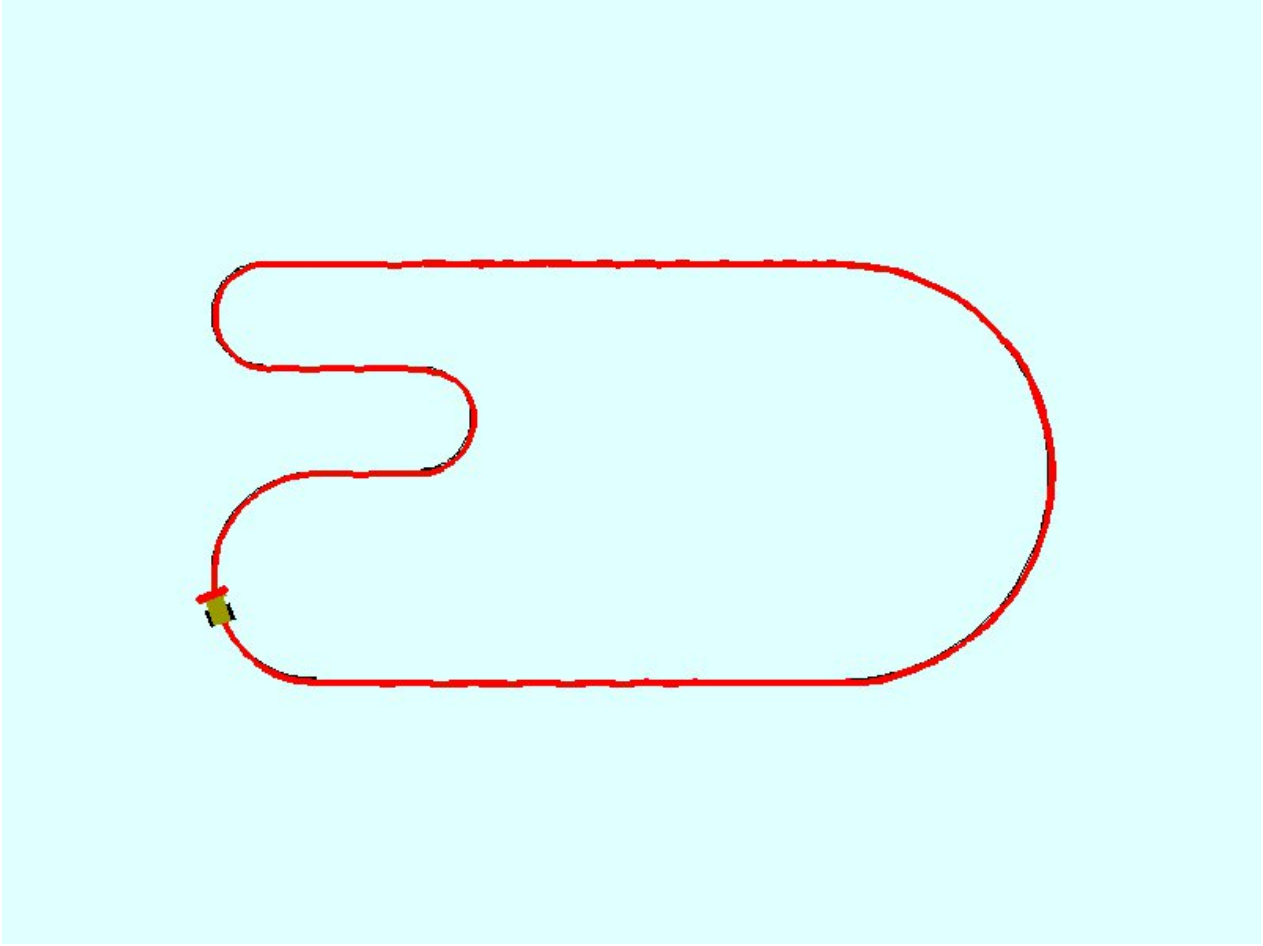

Figura 1: Resultado do caminho do seguidor de linha com Sarsa

4 Q-Leaning

4.1 Código

```python
class QLearning(RLAlgorithm):
    def __init__(self, num_states, num_actions, epsilon, alpha, gamma):
        super().__init__(num_states, num_actions, epsilon, alpha, gamma)

    def get_greedy_action(self, state):
        return epsilon_greedy_action(self.q,state,self.epsilon)

    def learn(self, state, action, reward, next_state, next_action):
        self.q[state][action]+=self.alpha*(reward
                            + self.gamma*np.max(self.q[next_state])- self.q[state][action])
```

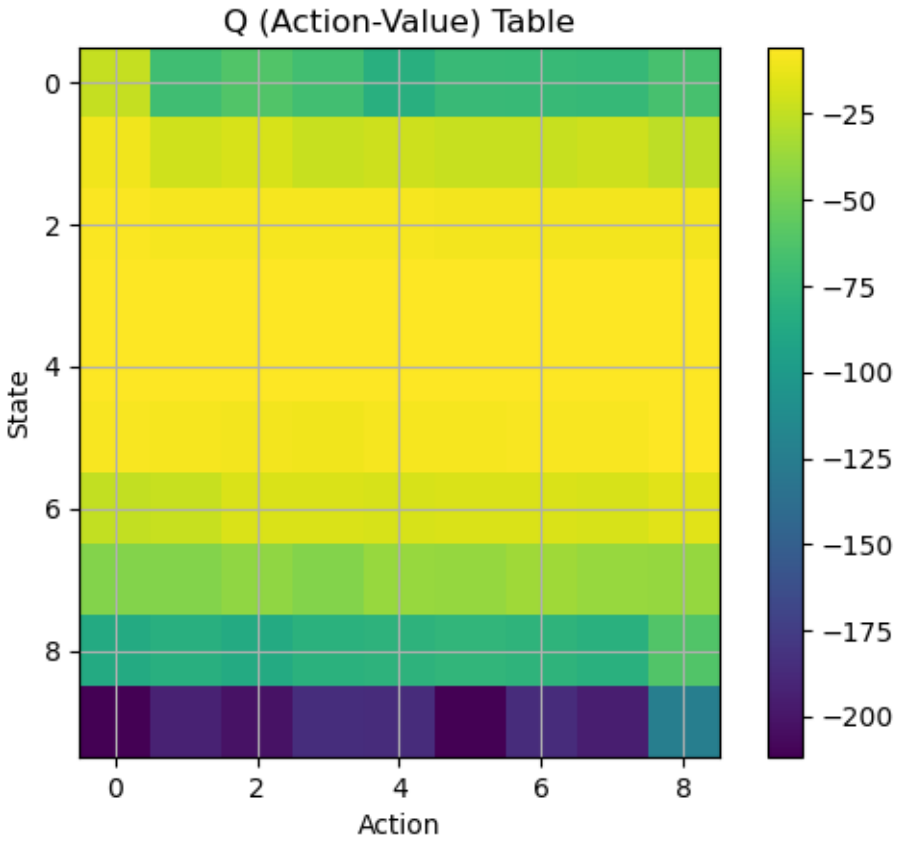

Figura 2: Tabela de ação valor com Sarsa

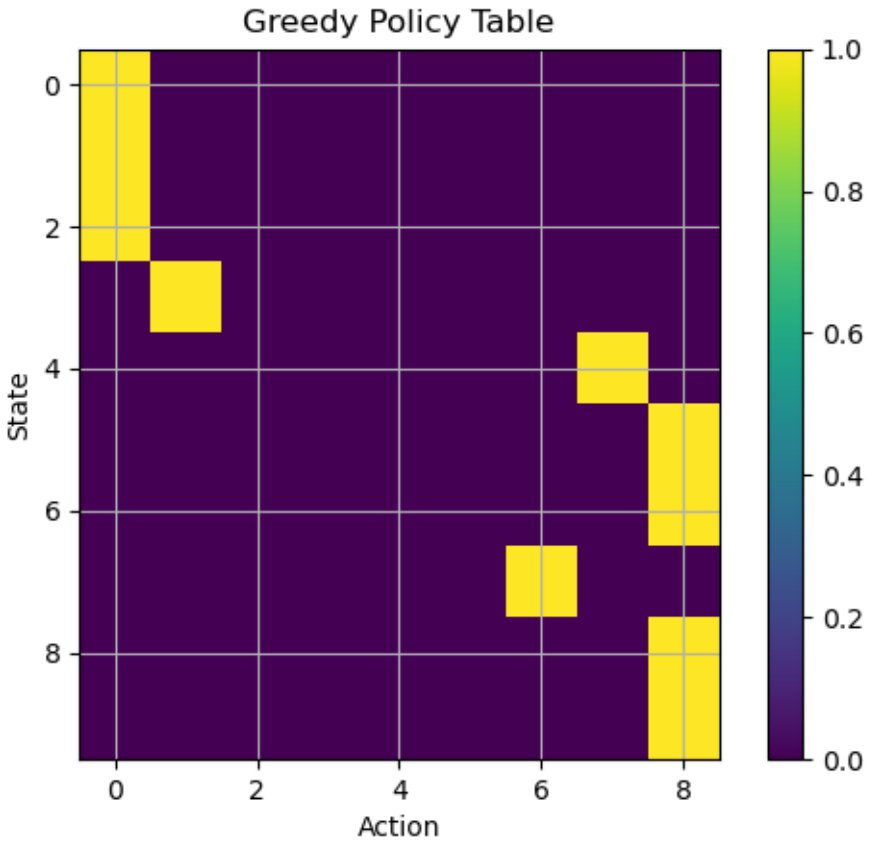

Figura 3: Tabela de políticas gulosas com Sarsa

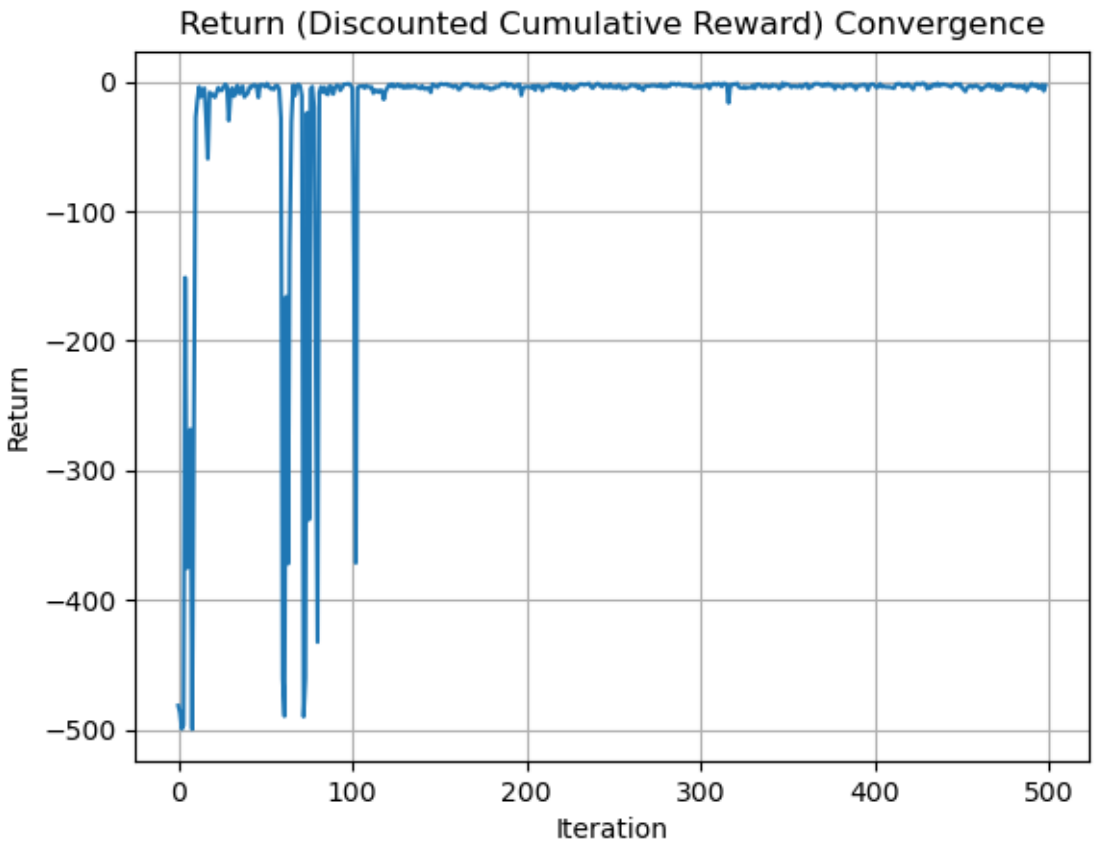

Figura 4: Convergência a cada iteração com Sarsa

4.2 Resultados e discussão

O resultado do arquivo de teste está mostrado abaixo:

```
Action-value Table:
[[-1.99       -1.         -2.9701    ]
 [-2.9663497  -1.99       -3.93423681]
 [-3.33507216 -2.9701     -3.82046584]
 [-4.27265574 -3.94039821 -5.13605225]
 [-5.1294525  -4.89546422 -4.89509086]
 [-4.37298888 -4.6636926  -3.94039894]
 [-3.49771597 -4.06441915 -2.9701    ]
 [-2.96309196 -3.93495926 -1.99      ]
 [-1.99       -2.9701     -1.        ]
 [ 0.         -0.99       -0.99      ]]
Greedy policy learnt:
[L, L, L, L, R, R, R, R, R, S]
```

O resultado, apesar de diferente do caso com o Sarsa, na realidade equivale a uma política equivalente: note que no centro do grid, tanto faz se mover para a direita ou para a esquera. Logo, ambos resultados eram esperados.

Fazendo o teste de convergência com o carrinho, os resultados estão mostrados nas figuras 5, 6, 7 e 8. Observe que novamente o carrinho conseguiu realizar o percurso proposto.

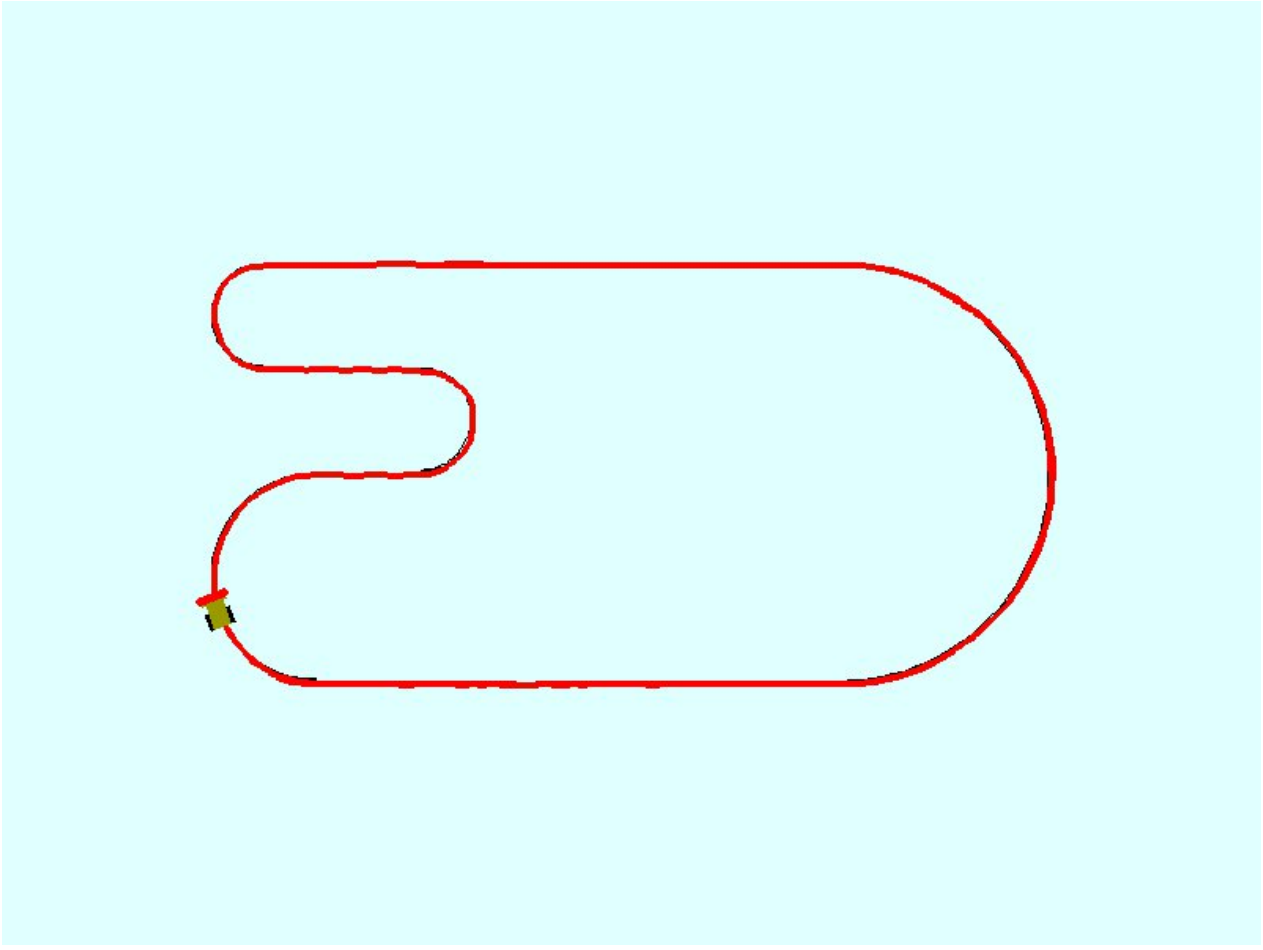

Figura 5: Resultado do caminho do seguidor de linha com Q-learning

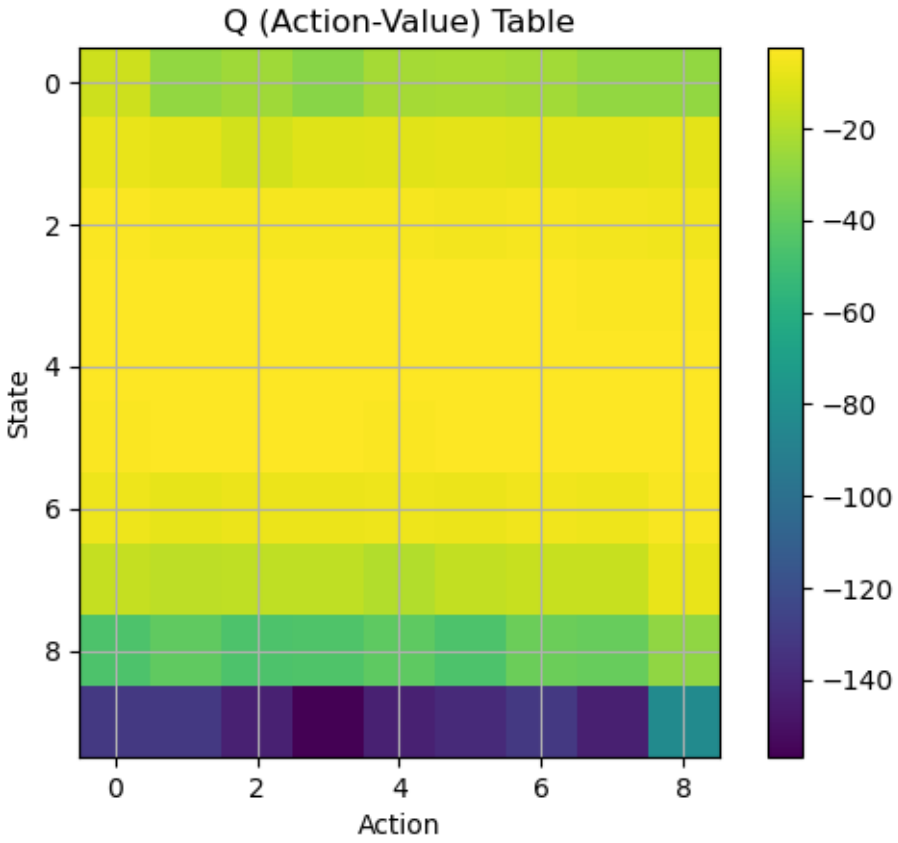

Figura 6: Tabela de ação valor com Q-Learning

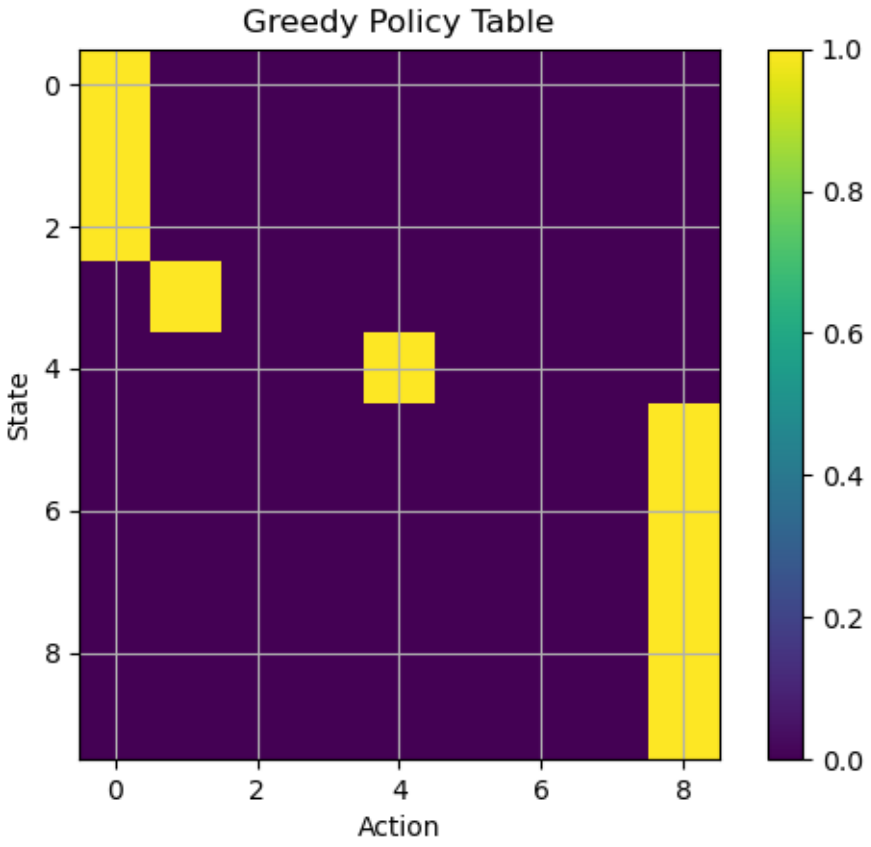

Figura 7: Tabela de políticas gulosas com Q-Learning

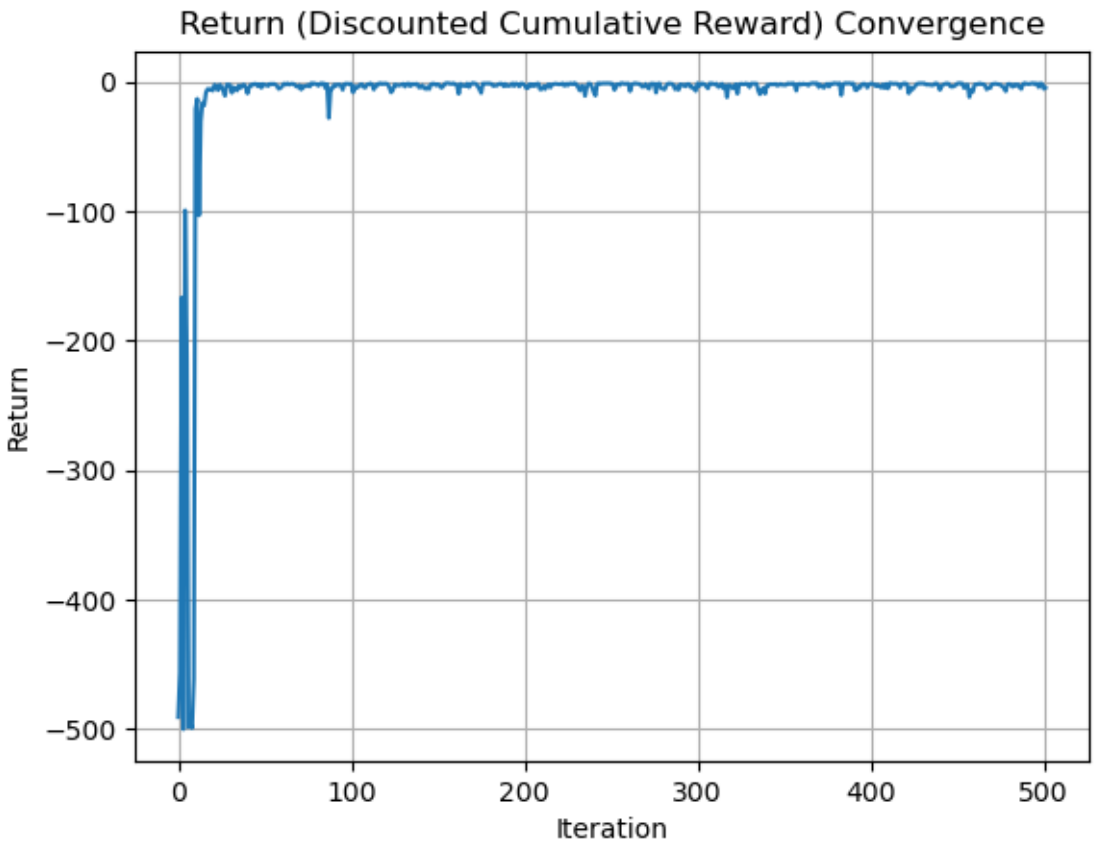

Figura 8: Convergência a cada iteração com Q-Learning