FOLDING BLOCKS – APRENDIZAGEM POR REFORÇO

TRABALHO Nº2 / TEMA 1B – TURMA 3MIEIC08 – IART

Nuno Marques (up20178997@fe.up.pt)

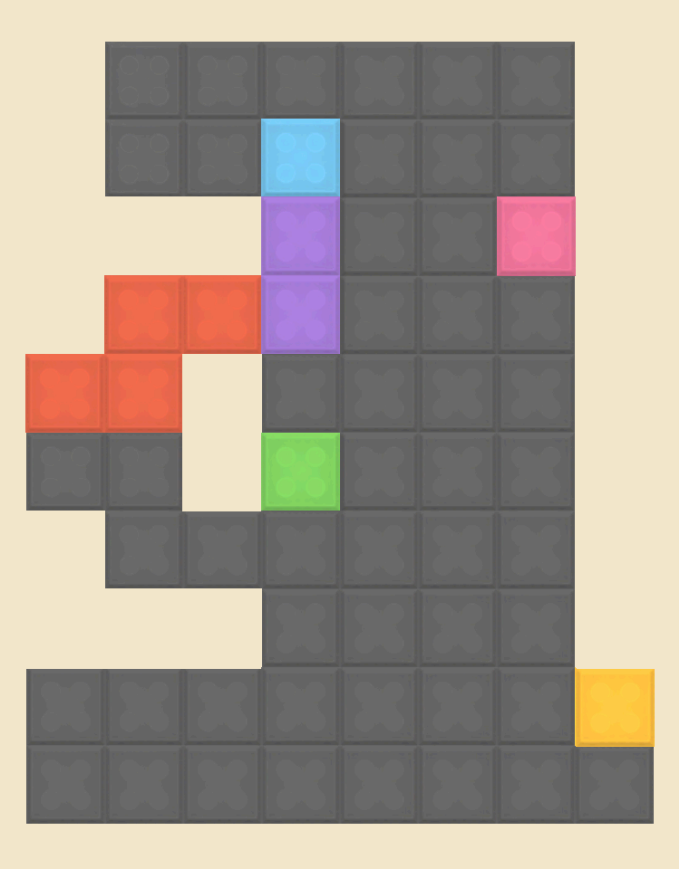
Ricardo Ferreira (up200305418@fe.up.pt)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

SUMÁRIO

- Especificação do problema de aprendizagem
- Descrição do problema
- Implementação
- Resultados
- Conclusões

ESPECIFICAÇÃO DO PROBLEMA



Folding Blocks

- Preencher o tabuleiro duplicando a área das peças de jogo
- Para cada peça, quatro acções possíveis (Duplicar para cima, duplicar para baixo, duplicar para a esquerda, duplicar para a direita)
- Peças não podem sair da área de jogo nem se sobrepor a outras peças
- Resolução optima obtida com o menor número de movimentos possível

DESCRIÇÃO DO PROBLEMA

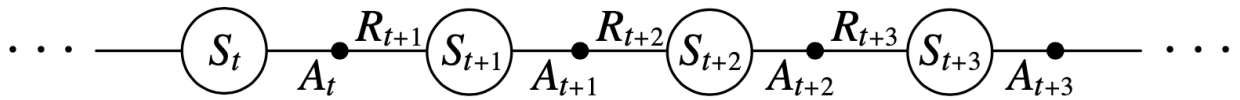
Objectivo: Ensinar um agente a solucionar os tabuleiros do jogo Folding Blocks utilizando técnicas de aprendizagem por reforço:

- Q-Learning
- SARSA

Os algoritmos de aprendizagem por reforço são constituidos pelos seguintes elementos:

- Policy (π) - Define como o agente deve agir e escolher as acções
- Reward (*r*) - Define o objectivo da aprendizagem. O ambiente recompensa o agente por cada acção que ele toma e este por sua vez tenta maximizar a recompensa.
- Value function (*v*) - Permite calcular o valor de cada estado, ou seja, o total de recompensa que se espera receber a partir do estado actual.

A evolução dos diferentes estados é representada por:



FORMULAÇÃO DO PROBLEMA

Estados: Tabuleiros de jogo com as peças após jogada.

Acções: (Por cada peça e em cada estado)

- DBUP: Duplicar peça para cima
- DBDWN: Duplicar peça para baixo
- DBLFT: Duplicar peça para a esquerda
- DBRGT: Duplicar peça para a direita

Recompensas:

- Escolha de acção válida: -1
- Escolha de acção inválida: -10
- Atingido estado em que não é possível terminar o jogo: -100
- Atingido estado final de sucesso: 100

IMPLEMENTAÇÃO

Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max\alpha, Q(s_{t+1}, a_t) - Q(s_t, a_t)]$$

```
Initialize Q(s,a) (all values = 0)
for(each episode)
    Initialize s
    for (each step of the episode,
    until s is terminal)
        choose a from s using policy
        from Q (e.g. greedy)
        take action a, observe r and s'
        update qTable using the equation
```

SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

```
Initialize Q(s,a) (all values = 0)
for(each episode)
    Initialize s
    choose a from s using policy
        from Q (e.g. greedy)
    for (each step of the episode,
    until s is terminal)
        take action a, observe r and s'
        choose a' from s' using policy
        from Q (e.g. greedy)
        update qTable using equation
```

Utilização de um parâmetro **ε** (entre 0 e 1) que permite previligear políticas mais aleatórias ou mais baseadas na tabela

IMPLEMENTAÇÃO

Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max\alpha, Q(s_{t+1}, a_t) - Q(s_t, a_t)]$$

```
Initialize Q(s,a) (all values = 0)
for(each episode)
    Initialize s
    for (each step of the episode,
    until s is terminal)
        choose a from s using policy
        from Q (e.g. greedy)
        take action a, observe r and s'
        update qTable using the equation
```

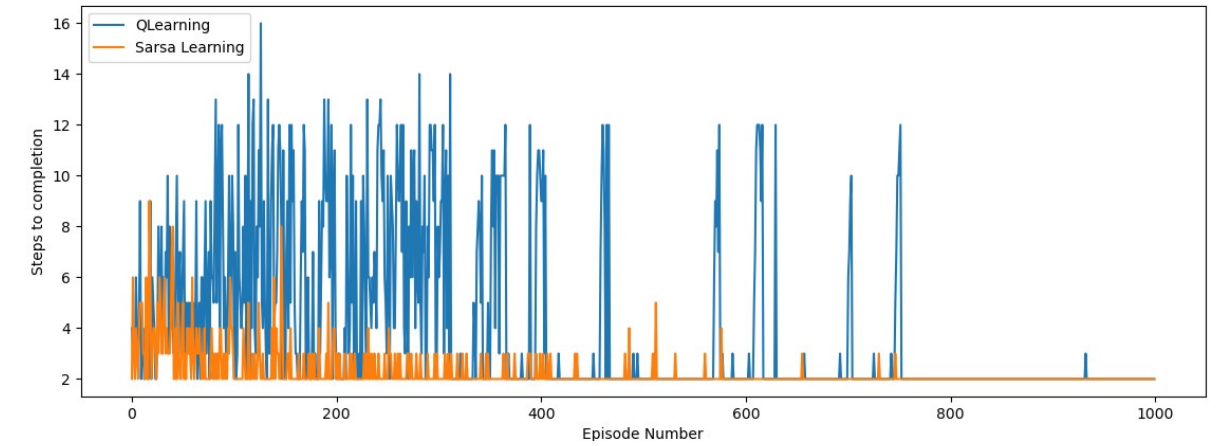
SARSA

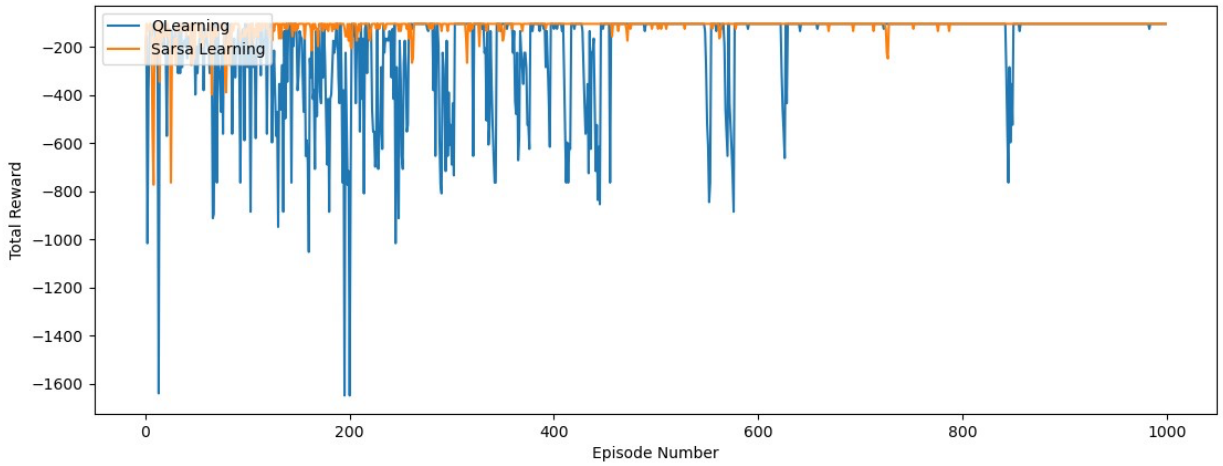
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

```
Initialize Q(s,a) (all values = 0)
for(each episode)
    Initialize s
    choose a from s using policy
        from Q (e.g. greedy)
    for (each step of the episode,
    until s is terminal)
        take action a, observe r and s'
        choose a' from s' using policy
        from Q (e.g. greedy)
        update qTable using equation
```

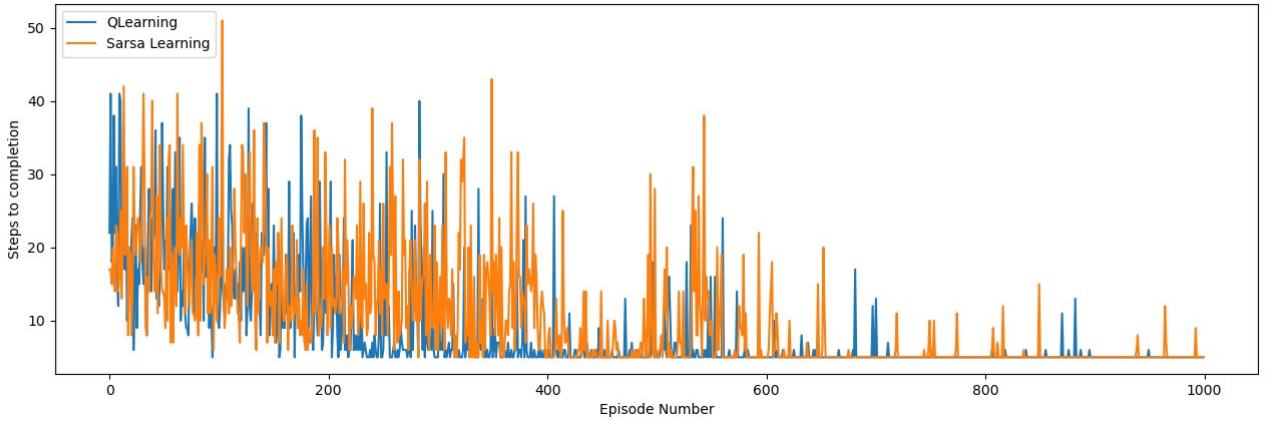
Utilização de um parâmetro **ε** (entre 0 e 1) que permite previligear políticas mais aleatórias ou mais baseadas na tabela

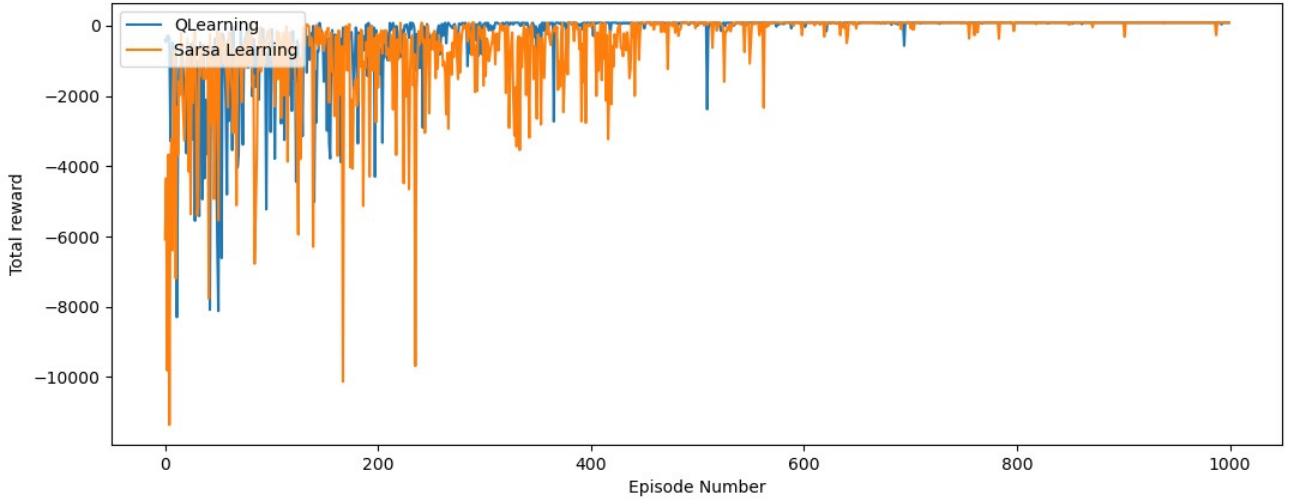
RESULTADOS (NÍVEL FÁCIL)



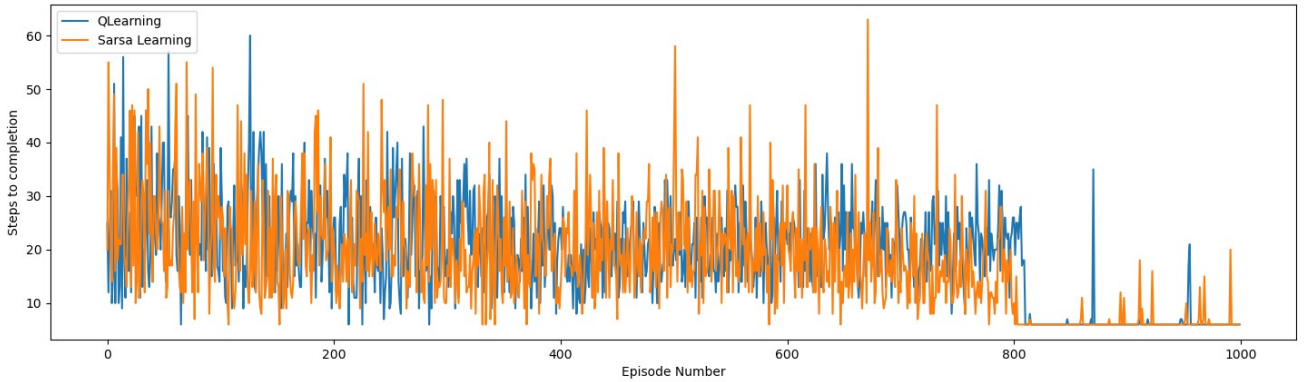


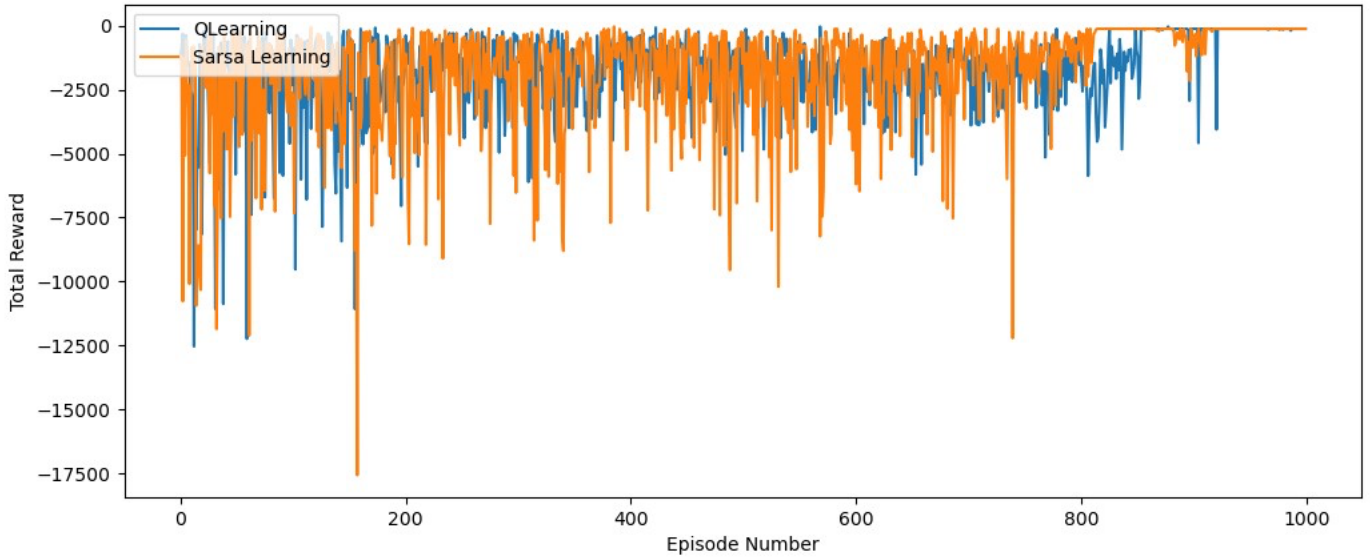
RESULTADOS (NÍVEL INTERMÉDIO)





RESULTADOS (NÍVEL DIFÍCIL)





CONCLUSÕES

- Os algoritmos escolhidos são válidos para ensinar um agente a jogar o jogo proposto
- Os algoritmos em análise apresentam resultados muito semelhantes
- O algoritmo SARSA é ligeiramente melhor, convergindo para a solução final mais rapidamente, especialmente em ambientes mais complexos
- O algoritmo Q-Learning apesar de convergir para o resultado final apresenta um evolução mais instável
- Quando a política de escolhas de acções é mais aleatória essa aleatoriedade é passada para o processo de aprendizagem