Universidade Federal de Minas Gerais

Inteligência Artificial
Mestrado em Ciência da Computação

Aprendizado por Reforço [Pac-Maze] RELATÓRIO

Autor:
Daniel Martins Reis

Matrícula:
2019697267

1 Pac-Maze

O presente trabalho consiste em implementar e comparar um labirinto simplificado do Jogo Pac-Man chamado Pac-Maze. O objetivo geral é exercitar o básico sobre aprendizado por reforço apresentado durante o curso da disciplina de Inteligência Artificial ministrado pelo professor Luiz Chaimowicz.

O cenário é um mundo bidimensional, representado por uma matriz de caracteres. A pastilha é representada por 0 (zero), um fantasma por & (e comercial), uma parede por #, e um espaço vazio por -.

2 Implementação

2.1 Modelagem

O mundo é modelado como um MDP <S, A, R, T> com as seguintes características:

- S: conjunto de estados são as posições onde o agente pode estar (-, 0 ou &);
- A: conjunto de ações: acima (U), abaixo (D), esquerda (L) e direita (R);
- R: a função de recompensa é a seguinte: em - a recompensa é -1, em 0 a recompensa é 10 (pílula!) e em & a recompensa é -10 (fantasma!). Isso vai incentivar o Pac-Man a encontrar o menor caminho até a pílula, evitando fantasmas;
- T: para facilitar, a função de transição é determinística: o agente consegue se mover na direção desejada. Por exemplo, a ação "U" em (3,1) leva o agente a (2,1). Se tentar se mover para uma parede, o Pac-Man não se desloca e recebe (novamente) a recompensa do estado onde está.

Dessa forma, o mundo foi armazenado por meio de uma matriz contendo em cada posição um valor (#, -, 0 ou &) que referem-se respectivamente a (parede, espaço vazio, pílula e fantasma); as ações por um vetor ['L','R','U','D'] e a Q-Table como um dicionário do python, contendo uma tupla (estado [x,y] na matriz, ação, movimento realizado e recompensa).

2.2 Carregando a instância do mundo

Lê as linhas do arquivo de entrada, identifica as dimensões do mundo, cria uma matriz de tamanho das dimensões e preenche a matriz com as informações lidas.

2.3 Posicionando o agente num local aleatório

Gera uma posição aleatória e verifica se ela é válida (ou seja, é um espaço vazio). Dessa forma, faz isso até que escolha uma posição válida.

2.4 Retornando o q-valor para um par estado x ação

Retorna o Q-valor para um par estado x ação da tabela ou zero, simplesmente porque a Q-Table é um dicionário do python, ou seja, o elemento procurado pode não está lá. Caso não esteja, o valor retornado é zero.

2.5 Retornando uma ação escolhida

Gera um valor aleatório entre 0 e 1. Se ele for menor que um epsilon, explora, ou seja, apenas retorna uma ação aleatória dentre as possiveis ações. Senão, exploita. No processo de exploitar, o que se faz é armazenar em q os valores da qTable para a ação solicitada e as possíveis ações a serem realizadas a partir desse estado, e a partir disso encontrar o maior q. Nisso, temos dois casos (empate ou não), ou seja, duas ou mais opções podem ser a melhor. Quando temos empate, o que se faz é escolher aleatoriamente dentre as ações empatadas.

2.6 Calculando a recompensa

Seleciona o elemento na posição determinada e retorna a recompensa de acordo com o tipo da célula (Viver no mundo, pílula ou fantasma)

- REWARD_PILL = 10 => Recompensa da pílula;
- REWARD_GHOST = -10 => Recompensa do fantasma;
- REWARD_SPACEFREE = -1 => Recompensa do espaço vazio (viver no mundo).

2.7 Aprendendo

Dado um estado 1, uma ação 1 e uma recompensa, o agente aprende qual a ação ele deve tomar. Para isso, guarda o Q-valor máximo entre as ações do estado 2 e o Q-valor atual para o estado 1 com uma ação 1 ou nulo caso não exista e calcula o novo Q-valor, de forma que se o Q-valor atual for nulo, retorna o valor da recompensa imediata. Senão, calcula o novo valor poderado. Por fim, atualiza o valor do novo Q-valor para aquele par estado e ação.

2.8 Movimentando o agente

Armazena o índice da posição atual e analisa o movimento escolhido (esquerda, direita, cima ou baixo). Caso seja possível realizá-lo (a posição seja válida e não seja uma parede, realiza o movimento e retorna True. Caso contrário, retorna False.

2.9 Ordenando, salvando e exibindo a Q-table

Ordena a q-table pelo valor de x,y, atualiza a q-table com a ordenação, agrupa os valores no formato (h, w, moviment, qvalue) em que:

- 'h' representa a posição na altura da matriz;
- 'w' representa a posição na na largura da matriz;
- 'moviment' representa o movimento realizado;
- 'qvalue' representa o Q-Valor na tabela.

Por fim, salva os dados agrupados no arquivo de saída (denominado "q.txt").

2.10 Computando, salvando e exibindo a política ótima

Cria a matriz de política a partir de um clone do mundo, em que percorre cada posição do mundo e verifica:

- Se for uma parede, preenche com (valor, INFINITO);
- Se não preenche com (espaço livre, -INFINITO).

A utilização do INFINITO E -INFINITO se justifica para permitir posteriormente computar o menor elemento. Portanto, percorre a Q-Table, e se o q-valor anterior (presente na política) for maior que o atual, atualiza a política para o atual, pois ele

é maior, já que o objetivo é maximizar a recompensa esperada. Nesse processo, os dados da política ótima obtida são salvos no arquivo de saída (denominado "pi.txt").

2.11 Executando um episódio

Cada episódio, consiste em setar o agente numa posição aleatória válida e fica num loop:

- Calcula a recompensa imediata da posição atual;
- Seleciona uma ação a partir do estado atual;
- Realiza movimento do agente => realiza a ação escolhida caso seja possível e aprende;

Vale ressaltar, que o que marca o fim de um episódio é o ato de chegar ao estado terminal. Dessa forma, os episódios são realizados n vezes com os parâmetros lidos determinados (alpha, gamma, epsilon, n).

3 Testes de execução

Os resultados descritos posteriormente foram obtidos executando com as seguintes entradas:

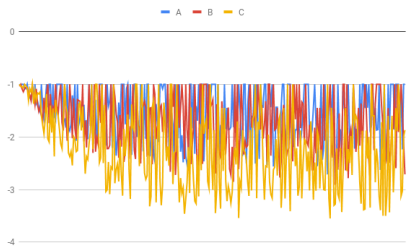
- "pacmaze.txt";
- "pacmaze-01-tiny.txt";
- "pacmaze-02-mid-sparse.txt";
- "pacmaze-03-tricky.txt".

Para cada arquivo, foram utilizadas as combinações geradas pelas seguintes configurações:

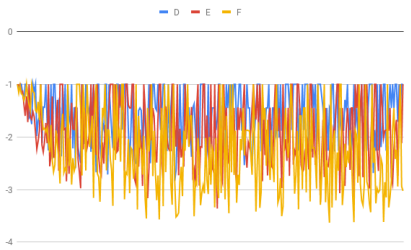
- Alpha: [0.1, 0.3, 0.5, 0.9]
- Epsilon: [0.1, 0.5, 0.9]
- Gamma fixo = 0.9
- N = Número de execuções => varia de acordo com a instância

4 Resultados

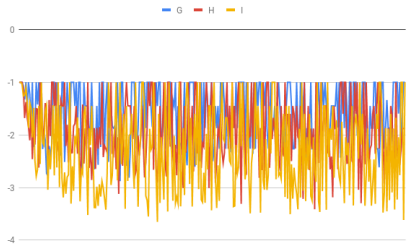
4.1 Entrada: pacmaze.txt



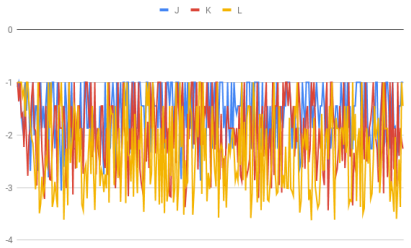
(a) Execução A, B, C



(b) Execução D, E, F



(c) Execução G, H, I



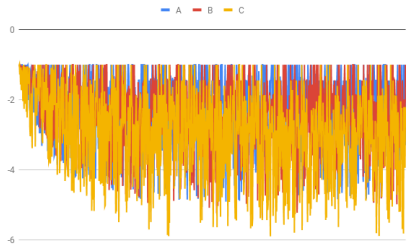
(d) Execução J, K, L

Figura 1: Recompensa média por episódio

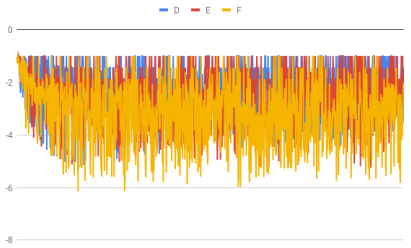
Legenda:

	Nome do arquivo	Alpha	E-greedy	Gamma	N
A	pacmaze.txt	0.1	0.1	0.9	300
B	pacmaze.txt	0.5	0.1	0.9	300
C	pacmaze.txt	0.9	0.1	0.9	300
D	pacmaze.txt	0.1	0.3	0.9	300
E	pacmaze.txt	0.5	0.3	0.9	300
F	pacmaze.txt	0.9	0.3	0.9	300
G	pacmaze.txt	0.1	0.5	0.9	300
H	pacmaze.txt	0.5	0.5	0.9	300
I	pacmaze.txt	0.9	0.5	0.9	300
J	pacmaze.txt	0.1	0.9	0.9	300
K	pacmaze.txt	0.5	0.9	0.9	300
L	pacmaze.txt	0.9	0.9	0.9	300

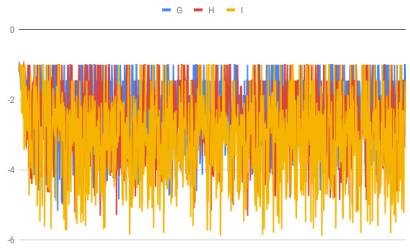
4.2 Entrada: pacmaze-01-tiny.txt



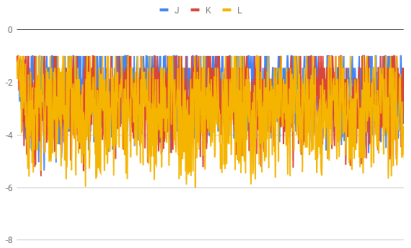
(a) Execução A, B, C



(b) Execução D, E, F



(c) Execução G, H, I



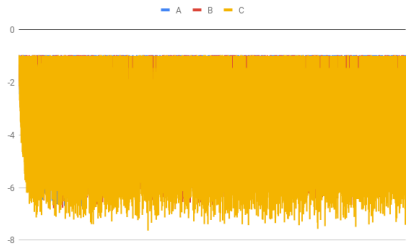
(d) Execução J, K, L

Figura 2: Recompensa média por episódio

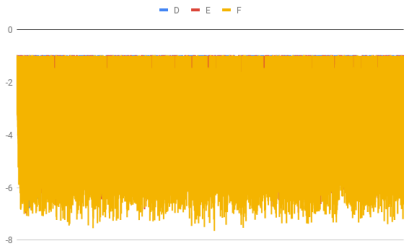
Legenda:

	Nome do arquivo	Alpha	E-greedy	Gamma	N
A	pacmaze-01-tiny.txt	0.1	0.1	0.9	1000
B	pacmaze-01-tiny.txt	0.5	0.1	0.9	1000
C	pacmaze-01-tiny.txt	0.9	0.1	0.9	1000
D	pacmaze-01-tiny.txt	0.1	0.3	0.9	1000
E	pacmaze-01-tiny.txt	0.5	0.3	0.9	1000
F	pacmaze-01-tiny.txt	0.9	0.3	0.9	1000
G	pacmaze-01-tiny.txt	0.1	0.5	0.9	1000
H	pacmaze-01-tiny.txt	0.5	0.5	0.9	1000
I	pacmaze-01-tiny.txt	0.9	0.5	0.9	1000
J	pacmaze-01-tiny.txt	0.1	0.9	0.9	1000
K	pacmaze-01-tiny.txt	0.5	0.9	0.9	1000
L	pacmaze-01-tiny.txt	0.9	0.9	0.9	1000

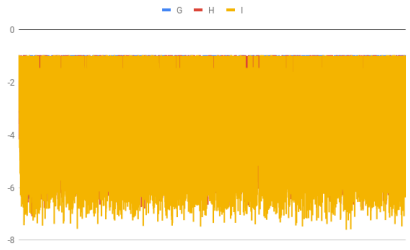
4.3 Entrada: pacmaze-02-mid-sparse.txt



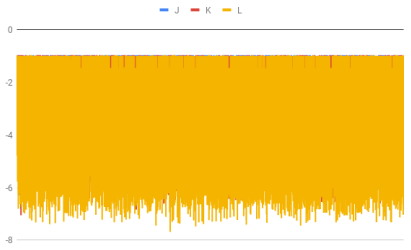
(a) Execução A, B, C



(b) Execução D, E, F



(c) Execução G, H, I



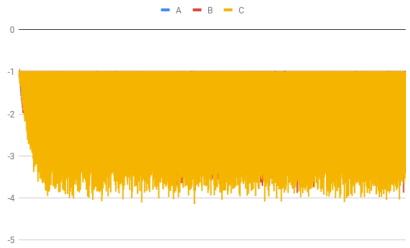
(d) Execução J, K, L

Figura 3: Recompensa média por episódio

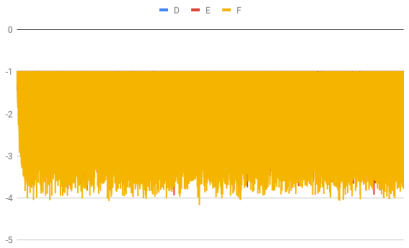
Legenda:

	Nome do arquivo	Alpha	E-greedy	Gamma	N
A	pacmaze-02-mid-sparse.txt	0.1	0.1	0.9	20000
B	pacmaze-02-mid-sparse.txt	0.5	0.1	0.9	20000
C	pacmaze-02-mid-sparse.txt	0.9	0.1	0.9	20000
D	pacmaze-02-mid-sparse.txt	0.1	0.3	0.9	20000
E	pacmaze-02-mid-sparse.txt	0.5	0.3	0.9	20000
F	pacmaze-02-mid-sparse.txt	0.9	0.3	0.9	20000
G	pacmaze-02-mid-sparse.txt	0.1	0.5	0.9	20000
H	pacmaze-02-mid-sparse.txt	0.5	0.5	0.9	20000
I	pacmaze-02-mid-sparse.txt	0.9	0.5	0.9	20000
J	pacmaze-02-mid-sparse.txt	0.1	0.9	0.9	20000
K	pacmaze-02-mid-sparse.txt	0.5	0.9	0.9	20000
L	pacmaze-02-mid-sparse.txt	0.9	0.9	0.9	20000

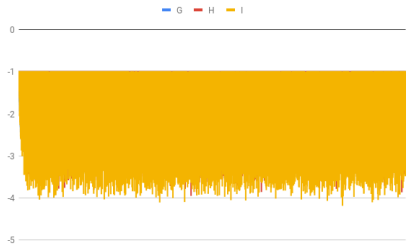
4.4 Entrada: pacmaze-03-tricky.txt



(a) Execução A, B, C



(b) Execução D, E, F



(c) Execução G, H, I

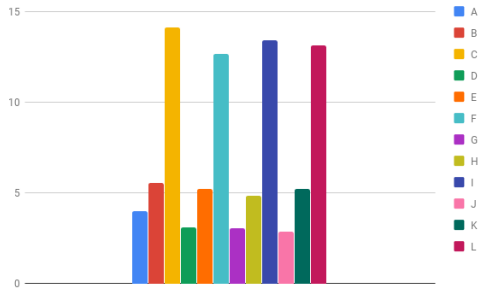
(d) Execução J, K, L

Figura 4: Recompensa média por episódio

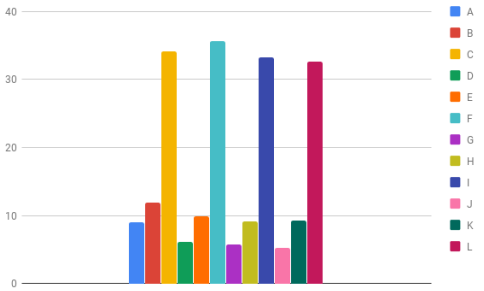
Legenda:

	Nome do arquivo	Alpha	E-greedy	Gamma	N
A	pacmaze-03-tricky.txt	0.1	0.1	0.9	25000
B	pacmaze-03-tricky.txt	0.5	0.1	0.9	25000
C	pacmaze-03-tricky.txt	0.9	0.1	0.9	25000
D	pacmaze-03-tricky.txt	0.1	0.3	0.9	25000
E	pacmaze-03-tricky.txt	0.5	0.3	0.9	25000
F	pacmaze-03-tricky.txt	0.9	0.3	0.9	25000
G	pacmaze-03-tricky.txt	0.1	0.5	0.9	25000
H	pacmaze-03-tricky.txt	0.5	0.5	0.9	25000
I	pacmaze-03-tricky.txt	0.9	0.5	0.9	25000
J	pacmaze-03-tricky.txt	0.1	0.9	0.9	25000
K	pacmaze-03-tricky.txt	0.5	0.9	0.9	25000
L	pacmaze-03-tricky.txt	0.9	0.9	0.9	25000

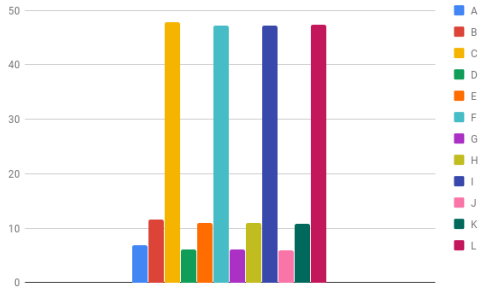
4.5 Tempo de vida para as 4 entradas



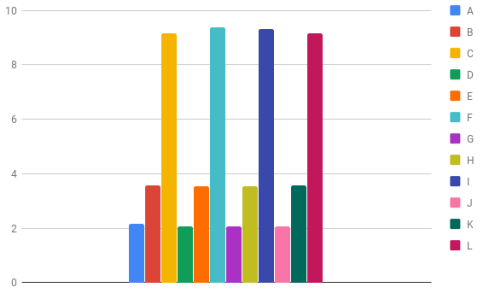
(a) Execução: pacmaze.txt



(b) Execução: pacmaze-01-tiny.txt



(c) Execução: pacmaze-02-mid-sparse.txt



(d) Execução: pacmaze-03-tricky.txt

Figura 5: Tempo de vida médio entre os episódios realizados para cada tipo de execução

5 Discussão dos Resultados

Nota-se que para a execução de todas as entradas, o fator alpha e E-greedy impactam diretamente na forma como a agente aprende. Dessa maneira, nota-se que um fator ideal é um maior alpha (mais próximo de 1) e um menor E-greedy (mais próximo de 0). Isso faz com que o agente explore mais uma melhor solução encontrada. Consequentemente, exploite menos, ou seja, realize menos vezes a ação de encontrar novos caminhos. O fato de se identificar que o fator alpha deve estar mais próximo de 1 num melhor cenário, é explicado pelo fato desse fator representar a taxa de aprendizagem do agente.

De forma análoga, nota-se que com um maior alpha e um menor E-greedy o agente vive na média por mais tempo no mundo no decorrer de cada episódio, isso porque ele

aprende e explora mais. Portanto, C, F, I e L são melhores, ou seja, morrem menos e conseguem obter uma maior recompensa na média. Esse fato é representado pelos gráficos de tempo de sobrevivência em cada iteração. Outra observação de suma importância é que ao variar a quantidade de iterações, nota-se que o efeito de se ter um maior alpha e um menor E-greedy torna-se mais expressivo, revelando um agente mais esperto (melhor).

Além disso, quando se tem um alpha muito alto, ou seja, muito próximo de 1, o valor do E-greedy não influencia tanto na performance do agente. O oposto disso já não é válido, ou seja, a partir somente do E-greedy não é possível afirmar nada.

Ressalta-se que tais características levantadas, ocorreram para todas as 4 instâncias utilizadas de teste.

Referências

Russell, Stuart and Peter Norvig (2009), *Artificial Intelligence: A Modern Approach*, 3rd edition. Prentice Hall Press, Upper Saddle River, NJ, USA.