Inteligência Artificial 2020/21
1º Projeto: Ricochet Robots

2 de Outubro de 2020

1 Introdução

O primeiro projeto da unidade curricular de Inteligência Artificial (IA) tem como objetivo desenvolver um programa em Python que resolva o problema Ricochet Robots utilizando técnicas de procura de IA.

2 Descrição do problema

O problema Ricochet Robots é baseado num jogo de tabuleiro com o mesmo nome, também conhecido por *Rasende Roboter* ou *Randolph's Robots*, que foi inventado na Alemanha em 1999 por Alex Randolph.

O jogo decorre sobre um tabuleiro com uma **grelha quadrada**. Cada célula da grelha pode ter **barreiras** nas bordas superior, inferior, esquerda ou direita. O tabuleiro é limitado por barreiras em toda a sua volta, i.e., cada célula na linha do topo tem uma barreira na borda superior, cada célula na coluna mais à direita tem uma barreira na borda da direita, etc.

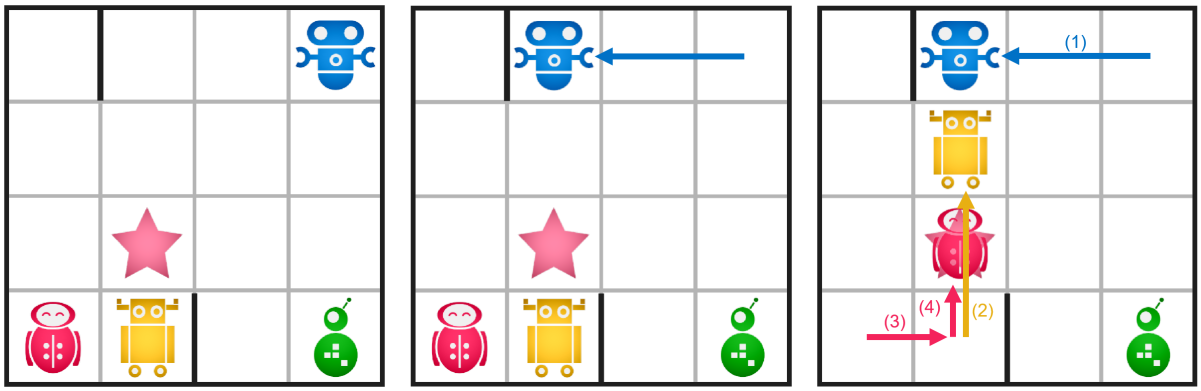
Além da grelha, o problema Ricochet Robots é composto por **4 robôs de cores diferentes**. No início do jogo, os robôs estão posicionados em quaisquer 4 células da grelha, tal que cada célula não pode conter mais do que um robô. Além dos robôs, na grelha está também presente um **alvo**. O alvo está posicionado em qualquer célula da grelha, e tem a cor de um dos robôs. A Figura 1a mostra um exemplo da disposição inicial de um tabuleiro 4×4.

O **objetivo** do jogo é identificar uma sequência de jogadas, em que cada jogada corresponde a um movimento de um robô, tal que, após a execução desta sequência de jogadas, o robô com a cor do alvo esteja posicionado na mesma célula que contém o alvo.

Os robôs podem mover-se sobre a grelha de acordo com as seguintes **regras**:

- Um robô pode movimentar-se em qualquer das 4 direções — cima, baixo, esquerda e direita — mas não na diagonal.
- Quando um robô se movimenta, só pára quando embate (i) numa barreira ou (ii) noutro robô.

A Figura 1b ilustra a configuração de um tabuleiro com um alvo vermelho após ser efetuado o movimento *robô azul para a esquerda*. A Figura 1c mostra uma possível configuração objetivo deste tabuleiro com o robô vermelho posicionado na mesma célula que o alvo da



(a) Tabuleiro inicial.

(b) Tabuleiro após um movimento.

(c) Objetivo atingido após 4 movimentos.

Figura 1: Resolução de uma instância de Ricochet Robots

mesma cor. Esta configuração objetivo pode ser obtida executando 4 movimentos sobre o tabuleiro original:

1) *robô azul para a esquerda;*
2) *robô amarelo para cima;*
3) *robô vermelho para a direita;*
4) *robô vermelho para cima.*

3 Objetivo

O objetivo deste projeto é o desenvolvimento de um programa em Python 3.8 que, dada uma instância de Ricochet Robots, retorna uma solução, i.e., uma sequência de movimentos dos robôs que resultem numa disposição em que o alvo e o robô da mesma cor ocupem a mesma célula na grelha. O programa deve ser desenvolvido num ficheiro `ricochet_robots.py`, que recebe como argumento da linha de comandos o caminho para um ficheiro que contém um instância de Ricochet Robots no formato descrito na secção 4.1. O programa deve resolver o problema utilizando uma técnica de procura à escolha e imprimir a solução para o *standard output* no formato descrito na secção 4.2.

Utilização:

```
python3 ricochet_robots.py <instance_file>
```

4 Formato de input e output

Nos ficheiros de input e output, os robôs são representados por uma letra maiúscula associada à sua cor (em Inglês): vermelho / *red* (**R**), amarelo / *yellow* (**Y**), verde / *green* (**G**) e azul / *blue* (**B**). Os movimentos e barreiras são representados por letras minúsculas indicativas da sua direção (em Inglês): cima / *up* (**u**), baixo / *down* (**d**), esquerda / *left* (**l**) ou direita / *right* (**r**).

4.1 Formato do input

Os ficheiros de input representam instâncias do problema Ricochet Robots e seguem o seguinte formato:

- A primeira linha tem apenas um inteiro N que indica a dimensão do tabuleiro $N \times N$;
- As quatro linhas seguintes indicam a posição dos robôs. Cada linha tem uma letra (R, Y, G ou B) e dois inteiros i e j, $1 \leq i, j \leq N$, separados por um espaço. O par (i, j) define a posição inicial do robô (linha, coluna) com a cor da letra correspondente.
- Uma linha com uma letra (R, Y, G, B) e dois inteiros i e j separados por um espaço, que definem, respectivamente, a cor e a posição do alvo.
- Uma linha com um inteiro K que define o número de barreiras internas do tabuleiro.
- K linhas em que cada linha define uma barreira interna no tabuleiro. Cada linha contém dois inteiros i e j e uma letra (u, d, l, r), separados por um espaço. Cada uma destas K linhas especifica que na posição (i, j) existe uma barreira na borda superior, inferior, esquerda ou direita, respectivamente.

Exemplo

O ficheiro de input que descreve a instância da Figura 1a é o seguinte:

```
4
Y 4 2
G 4 4
B 1 4
R 4 1
R 3 2
2
1 1 r
4 2 r
```

4.2 Formato do output

O output do programa deve descrever uma solução para o problema de Ricochet Robots descrito no ficheiro de input, i.e., uma sequência de movimentos dos robôs tais que, após aplicados ao estado inicial do tabuleiro, o alvo e o robô da mesma cor ocupem a mesma célula no tabuleiro. O output deve seguir o seguinte formato:

- Uma linha com um número inteiro M que define o número total de movimentos na solução;
- M linhas, com duas letras separadas por um espaço cada uma, em que:
  - a primeira letra representa um robô (R, Y, G, B)
  - a segunda define a direção do movimento (u, d, l, r).

Exemplo

O output que descreve a solução da Figura 1c é:

```
4
B l
Y u
R r
R u
```

Nota: Pode haver mais do que uma solução para uma instância de Ricochet Robots. Como tal, o output obtido pela execução de uma implementação do projeto pode não ser igual ao apresentado. Qualquer output obtido é considerado correto desde que os movimentos dos robôs estejam descritos corretamente e levem a uma disposição em que o alvo e o robô da mesma cor ocupam a mesma célula na grelha do tabuleiro.

5 Implementação

Nesta secção é descrito o código que poderá ser usado no projeto e o código que deverá ser implementado no projeto.

5.1 Código a utilizar

Para a realização deste projecto devem ser utilizados os ficheiros a ser disponibilizados no site da unidade curricular com a implementação em *Python* dos algoritmos de procura [1]. O mais

[1] Este código é adaptado a partir do código disponibilizado com o livro *Artificial Intelligence: a Modern Approach* e que está disponível em `https://github.com/aimacode`.

importante é compreender para que servem e como usar as funcionalidades implementadas nestes ficheiros.

Estes ficheiros não devem ser alterados. Se houver necessidade de alterar definições incluídas nestes ficheiros, estas alterações devem ser feitas no ficheiro de código desenvolvido que contém a implementação do projeto.

5.1.1 Procuras

No ficheiro `search.py` estão implementadas as estruturas necessárias para correr os diferentes algoritmos de procura. Destacam-se:

- Classe `Problem`: Representação abstrata do problema de procura;
- Função `breadth_first_tree_search`: Procura em largura primeiro;
- Função `depth_first_tree_search`: Procura em profundidade primeiro;
- Função `greedy_search`: Procura gananciosa;
- Função `astar_search`: Procura A*.

5.1.2 Classe `RRState`

Esta classe representa os estados utilizados nos algoritmos de procura. O membro `board` armazena a configuração do tabuleiro a que o estado corresponde. Abaixo é apresentado o código desta classe, que não deve ser alterado.

```
class RRState:
    state_id = 0

    def __init__(self, board):
        self.board = board
        self.id = RRState.state_id
        RRState.state_id += 1

    def __lt__(self, other):
        """ Este método é utilizado em caso de empate na gestão da lista
        de abertos nas procuras informadas. """
        return self.id < other.id
```

5.2 Código a implementar

5.2.1 Classe `Board`

A classe `Board` é a representação interna de um tabuleiro de Ricochet Robots. A implementação desta classe e respectivos métodos é livre. Deve, no entanto, incluir um método

`robot_position` que recebe como argumento um caracter indicativo da cor de um dos robôs (R, Y, G ou B) e deve devolver um tuplo com dois inteiros (i, j) que correspondem às coordenadas do robô no tabuleiro. Este método será utilizado para fazer testes à restante implementação da classe.

```
class Board:
    """ Representação interna de um tabuleiro de Ricochet Robots. """

    def robot_position(self, robot: str):
        """ Devolve a posição atual do robô passado como argumento. """
        # TODO
        pass

    # TODO: outros metodos da classe
```

5.2.2 Função `parse_instance`

A função `parse_instance` é responsável por ler o tabuleiro descrito no ficheiro de input e devolver um objeto do tipo `Board` que o represente.

```
def parse_instance(filename: str) -> Board:
    """ Lê o ficheiro cujo caminho é passado como argumento e retorna
    uma instância da classe Board. """
    # TODO
    pass
```

5.2.3 Classe `RicochetRobots`

A classe `RicochetRobots` herda da classe `Problem` definida no ficheiro `search.py` do código a utilizar e deve implementar os métodos necessários ao seu funcionamento.

O método `actions` recebe como argumento um estado e retorna uma lista de ações que podem ser executadas a partir desse estado. O método `result` recebe como argumento um estado e uma ação, e retorna o resultado de aplicar essa ação a esse estado. Em ambos os métodos, uma ação corresponde a mover um robô numa das 4 direções possíveis. Cada ação é representada sob a forma de um tuplo com 2 caracteres, o primeiro indica a cor do robô a mover e o segundo a direção do movimento. Por exemplo, `('B', 'l')` representa a ação "robô azul para a esquerda".

Para suportar as procuras informadas, nomeadamente a procura gananciosa e a procura A*, deve desenvolver uma heurística que consiga guiar da forma mais eficiente possível estas procuras. A heurística corresponde à implementação do método `h` da classe `RicochetRobots`. Esta função recebe como argumento um `node`, a partir do qual se pode aceder ao estado atual em `node.state`.

De seguida é disponibilizado um protótipo da classe `RicochetRobots` que pode ser usado como base para a sua implementação.

```
class RicochetRobots(Problem):
    def __init__(self, board: Board):
        """ O construtor especifica o estado inicial. """
        # TODO
        pass

    def actions(self, state: RRState):
        """ Retorna uma lista de ações que podem ser executadas a
        partir do estado passado como argumento. """
        # TODO
        pass

    def result(self, state: RRState, action):
        """ Retorna o estado resultante de executar a 'action' sobre
        'state' passado como argumento. A ação a executar deve ser uma
        das presentes na lista obtida pela execução de
        self.actions(state). """
        # TODO
        pass

    def goal_test(self, state: RRState):
        """ Retorna True se e só se o estado passado como argumento é
        um estado objetivo. Deve verificar se o alvo e o robô da
        mesma cor ocupam a mesma célula no tabuleiro. """
        # TODO
        pass

    def h(self, node: Node):
        """ Função heuristica utilizada para a procura A*. """
        # TODO
        pass
```

5.2.4 Exemplos de utilização

De seguida, são apresentados alguns exemplos da utilização do código a desenvolver, assim como o respetivo output. Estes exemplos podem ser utilizados para testar a implementação. Considere que o ficheiro `i1.txt` se encontra na diretoria a partir da qual o código está a ser executado e que contém a instância descrita na secção 4.1.

Exemplo 1:

```
# Ler tabuleiro do ficheiro i1.txt:
board = parse_instance('i1.txt')

# Imprimir as posições dos robôs:
print(board.robot_position('Y'))
print(board.robot_position('G'))
print(board.robot_position('B'))
print(board.robot_position('R'))
```

Output:

```
(4, 2)
(4, 4)
(1, 4)
(4, 1)
```

Exemplo 2:

```
# Ler tabuleiro do ficheiro 'i1.txt':
board = parse_instance('i1.txt')

# Criar uma instância de RicochetRobots:
problem = RicochetRobots(board)

# Criar um estado com a configuração inicial:
initial_state = RRState(board)

# Mover o robô azul para a esquerda:
result_state = problem.result(initial_state, ('B', 'l'))

# Imprimir a posição do robô azul:
print(result_state.board.robot_position('B'))
```

Output:

```
(1, 2)
```

Exemplo 3:

```
# Ler tabuleiro do ficheiro "i1.txt":
board = parse_instance("i1.txt")

# Criar uma instância de RicochetRobots:
problem = RicochetRobots(board)

# Criar um estado com a configuração inicial:
s0 = RRState(board)

# Aplicar as ações que resolvem a instância
s1 = problem.result(s0, ('B', 'l'))
s2 = problem.result(s1, ('Y', 'u'))
s3 = problem.result(s2, ('R', 'r'))
s4 = problem.result(s3, ('R', 'u'))

# Verificar que o robô está no alvo:
print(problem.goal_test(s4))
print(s4.board.robot_position('R'))
```

Output:

```
True
(3, 2)
```

Exemplo 4:

```
# Ler tabuleiro do ficheiro "i1.txt":
board = parse_instance("i1.txt")

# Criar uma instância de RicochetRobots:
problem = RicochetRobots(board)

# Obter o nó solução usando a procura A*:
solution_node = astar_search(problem)
```

O valor de retorno das funções de procura é um objeto do tipo `Node`. Do nó de retorno podem ser retiradas as ações que levaram ao estado final que devem ser usadas para produzir o output desejado. Para tal, podem ser usados os membros `node.parent`, que é o nó precedente ao atual, e `node.action`, que indica a ação que levou ao estado atual.

6 Avaliação

A nota do projecto será baseada nos seguintes critérios:

- Execução correcta (80% - 16 val.). Estes 16 valores serão distribuídos da seguinte forma:
  - 12 valores - testes realizados via submissão no Mooshak;
  - 4 valores - testes adicionais realizados após o prazo de entrega.
- Relatório (20% - 4 val.).

7 Condições de realização e prazos

O projecto deve ser realizado em grupos de 2 alunos.

O código do projecto deve ser entregue obrigatoriamente por via electrónica até às 23:59 do dia 30 de Outubro de 2020, através do sistema Mooshak. Deverá ser submetido o ficheiro `ricochet_robots.py` contendo o código do seu projecto. O ficheiro de código deve conter em comentário, nas primeiras linhas, o número e o nome dos alunos.

A avaliação da execução do código do projecto será feita automaticamente através do sistema Mooshak, usando vários testes configurados no sistema. Os testes considerados para efeitos de avaliação podem incluir ou não os exemplos disponibilizados, além de um conjunto de testes adicionais. O tempo de execução de cada teste está limitado, bem como a memória utilizada. Só poderá efectuar uma nova submissão pelo menos 15 minutos depois da submissão anterior. Só são permitidas 10 submissões em simultâneo no sistema, pelo que uma submissão poderá ser recusada se este limite for excedido. Nesse caso tente mais tarde.

Duas semanas antes do prazo da entrega, serão publicadas na página da cadeira as instruções necessárias para a submissão do código no Mooshak. Apenas a partir dessa altura será possível a submissão por via electrónica. Até ao prazo de entrega poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada.

Deve produzir um relatório contendo um máximo de duas páginas de texto com fonte 12pt e espaçamento normal. Para além destas duas páginas, pode acrescentar ao relatório imagens, figuras e tabelas. O relatório deve conter os resultados obtidos executando uma procura em largura primeiro, uma procura em profundidade primeiro, uma procura gananciosa e uma procura A*. Os resultados devem conter o tempo de execução, o número de nós expandidos e o número de nós gerados. Para obter alguns destes valores, pode usar no código publicado a classe `InstrumentedProblem` e o exemplo da sua utilização que se encontra no fim do ficheiro `search.py`. Deve ser feita uma análise crítica dos resultados obtidos, comparando em termos de completude, eficiência e otimalidade os diferentes métodos testados. Deve também analisar a heurística implementada e eventualmente compará-la com outras heurísticas avaliadas.

8 Cópias

Projectos iguais, ou muito semelhantes, originarão a reprovação na disciplina e, eventualmente, o levantamento de um processo disciplinar. Os programas entregues serão testados em relação a soluções existentes na web. As analogias encontradas com os programas da web serão tratadas como cópias.