



Universidade do Minho
Escola de Engenharia

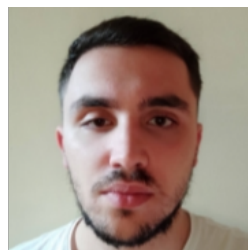
Inteligência Artificial 2022/2023

GRUPO 49

Artur Carneiro Neto de Nóbrega Luís A95414
Francisco Pinto Lameirão A97504
Lara Beatriz Pinto Ferreira A95454



A95414



A95454



A95454

1ª FASE
Dezembro de 2022

Índice

1	Introdução	2
2	Descrição do Problema	2
3	Formulação do Problema	3
3.1	Estado Inicial	3
3.2	Estado Final	3
3.3	Operadores	3
3.4	Custo Total da Solução	4
4	Menu	4
5	Geração do Circuito	5
6	Grafo	6
7	Estratégia Adotada	6
8	Conclusão	7

1 Introdução

Neste trabalho, proposto no âmbito da unidade curricular de *Inteligência Artificial*, desenvolveremos diversos algoritmos de procura para a resolução do jogo VectorRace, também conhecido como RaceTrack, de modo a conseguirmos resolver problemas através da conceção e implementação de algoritmos de procura.

Numa primeira fase, é pretendido gerar pelo menos um circuito VectorRace, com a indicação dos limites da pista, da posição inicial, que será o **I**, e da linha da meta, que corresponde ao **F**, em que o jogador terá de chegar ao **F** a partir de **I** da forma mais eficiente, seguindo assim, o caminho mais curto.

Para tal propósito, aplicamos o algoritmo de procura (não informada) BFS.

2 Descrição do Problema

Como referido anteriormente, o VectorRace, é um jogo de simulação de carros simplificado, que contém um conjunto de movimentos e regras associadas, de modo a um carro conseguir completar o circuito (deslocar-se do ponto I a F) percorrendo o caminho mais curto possível, evitando também bater nas "paredes" que contornam a pista.

O movimento do carro no VectorRace é bastante simples, tendo como base que as ações desenvolvidas são equivalentes a um conjunto de acelerações. Considerando a notação l , que representa a linha e c a coluna para os vetores, num determinado instante, o carro pode acelerar -1, 0 ou 1 unidades em cada direção (linha e coluna). Consequentemente, para cada uma das direções o conjunto de acelerações possíveis é $Acel = -1, 0, +1$, com $a = (a_l, a_c)$ a representar a aceleração de um carro nas duas direções num determinado instante.

Tendo em conta que p como tuplo que indica a posição de um carro numa determinada jogada j ($p_j = (p_l, p_c)$), e v o tuplo que indica a velocidade do carro nessa jogada ($v_j = (v_l, v_c)$), na seguinte jogada o carro irá estar na posição:

$$\begin{aligned} p_l^{(j+1)} &= p_l^j + v_l^j + a_l \\ p_c^{(j+1)} &= p_c^j + v_c^j + a_c \end{aligned}$$

A velocidade do carro num determinado instante é calculada:

$$\begin{aligned} v_l^{(j+1)} &= v_l^j + a_l \\ v_c^{(j+1)} &= v_c^j + a_c \end{aligned}$$

Sendo uma simulação de corrida de carros, existe a possibilidade de o carro sair da pista, sendo que quando essa situação ocorrer o carro terá de voltar para a posição anterior, assumindo um valor de velocidade zero.

Cada movimento de um carro numa determinada jogada, de uma determinada posição para outra, terá um custo de 1 unidade, sendo que quando o mesmo sair dos limites da pista, o custo é de 25 unidades.

3 Formulação do Problema

3.1 Estado Inicial

I representa a posição inicial do jogador, de onde o carro deve partir e iniciar a sua trajetória até à meta.

A velocidade é nula.

3.2 Estado Final

F representa a posição final do jogador, a meta. O objetivo do jogador é chegar a F.

A velocidade, nesta fase final, não é relevante para a vitória do jogador.

3.3 Operadores

Nome: Subir

Pré-condição: O espaço que o jogador pretende ocupar não ser parede ou obstáculo.

Efeito: O carro sobe uma linha.

Custo: 1

Nome: Descer

Pré-condição: O espaço que o jogador pretende ocupar não ser parede ou obstáculo.

Efeito: O carro desce uma linha.

Custo: 1

Nome: Deslocar para a esquerda

Pré-condição: O espaço que o jogador pretende ocupar não ser parede ou obstáculo.

Efeito: O carro passa para a coluna à esquerda da posição anterior.

Custo: 1

Nome: Deslocar para a direita

Pré-condição: O espaço que o jogador pretende ocupar não ser parede ou obstáculo.

Efeito: O carro passa para a coluna à direita da posição anterior.

Custo: 1

Nome: Colidir com parede ou obstáculo

Pré-condição: O espaço que o jogador pretende ocupar ser parede ou obstáculo.

Efeito: O carro volta para a posição anterior à colisão.

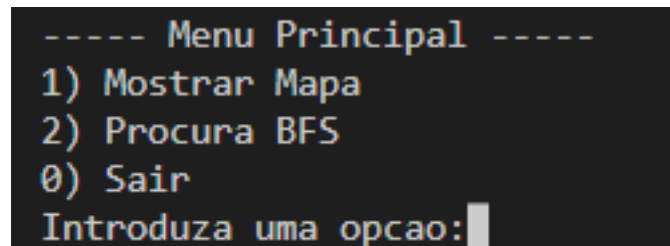
Custo: 25

3.4 Custo Total da Solução

O custo total da solução, será equivalente à soma de todos os movimentos que o carro faz para chegar à meta. Cada movimento tem o custo de 1 unidade. Se bater contra uma parede, o custo é de 25 unidades.

4 Menu

O nosso menu é composto pelas seguintes 3 opções:



```
----- Menu Principal -----
1) Mostrar Mapa
2) Procura BFS
0) Sair
Introduza uma opcao: |
```

Figura 1: Menu

- **1) Mostrar Mapa:** Imprime no terminal o circuito.
- **2) Procurar BFS:** Imprime no terminal a solução do circuito com cada etapa correspondente ao caminho mais curto do circuito (de I a F). Apresentando também o custo correspondente.
- **0) Sair:** Saímos do Menu.

5 Geração do Circuito

Neste momento, no nosso trabalho, é gerado apenas um mapa para o jogo. Selecionando a opção 1 ("Mostrar Mapa") do menu, apresentado anteriormente na *Figura 1*, é gerado o seguinte circuito:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	#	#	#	#	#	#	#	I	#	#	#	#	#	#	#
1	#	#	#	#	#						#	#	#	#	#
2	#	#												#	#
3	#						#	#	#						#
4	#				#	#	#	#	#	#	#				#
5	#	#	#			#	#	#	#					#	#
6	#	#				#	#	#				#	#	#	
7	#	#	#				#	#	#					#	#
8	#	#	#	#									#	#	#
9	#	#	#	#	#	#	#	F	#	#	#	#	#	#	#

Figura 2: Circuito

As dimensões do mapa são 10x15.

Cada símbolo, #, representa as paredes e obstáculos do circuito, e os espaços em branco revelam onde o carro pode se deslocar, do ponto de partida (I) até ao seu objetivo (F).

6 Grafo

Na construção do grafo, utilizamos duas funções: *constroiGrafo* e *expande*.

A função *constroiGrafo* é utilizada para criar a ligação no grafo entre o nodo atual e os nodos adjacentes que se encontram na lista que obtivemos na função *expande*, e para isso utiliza a função *adicionaAresta* que se encontra no *Grafo.py*.

A função *expande* é utilizada para expandir o estado atual, isto é, a função recebe o nodo atual e devolve uma lista com todos os nodos adjacentes que sejam viáveis para mover o carro na próxima jogada. Se o nodo adjacente for uma parede do circuito ou então estiver fora do mesmo, este não é adicionado à lista.

7 Estratégia Adotada

Estratégias de procura não informadas usam apenas as informações disponíveis na definição do problema.

O grupo decidiu implementar um algoritmo de procura não informado, **BFS** (Breath-First Search).

A figura seguinte apresenta a solução ótima e o seu custo, resultado da procura BFS implementada.

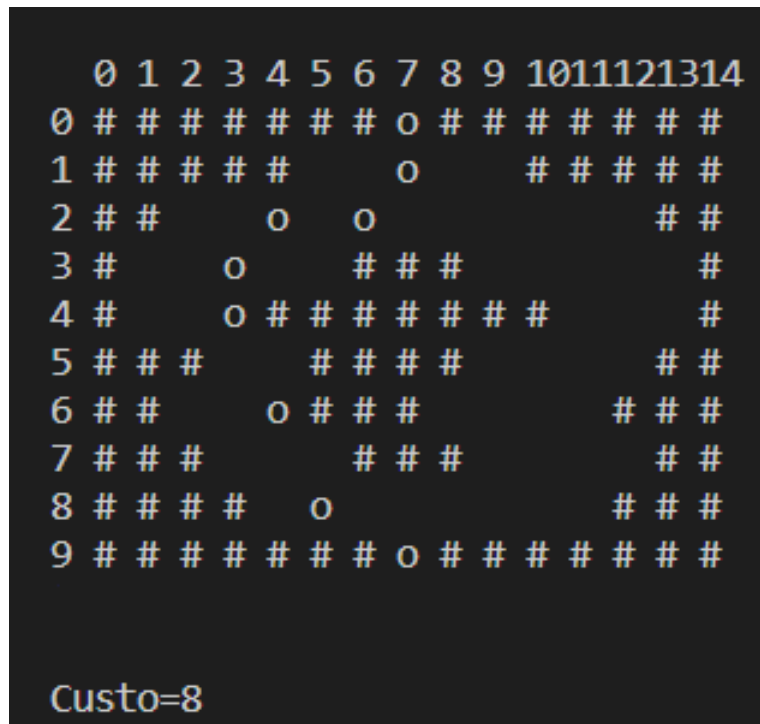


Figura 3: Procura BFS

8 Conclusão

O grupo está satisfeito com a realização desta primeira fase do trabalho e tudo que aprendeu com a mesma. Conseguimos consolidar o que aprendemos nas aulas de Inteligência Artificial e aprofundar esses mesmos conhecimentos na linguagem *python* e na resolução de problemas através da conceção e implementação de algoritmos de procura.

Para além do que a Unidade Curricular nos tem ensinado a nível técnico, também melhorou o grupo a ultrapassar dificuldades a nível de trabalhar em grupo, organizar tarefas e superar outras dificuldades inerentes à realização de um trabalho deste tipo.