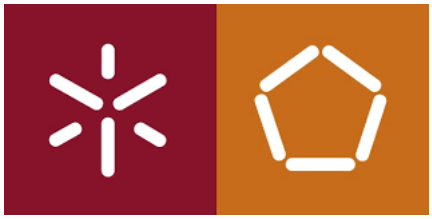

Universidade do Minho

Licenciatura em Engenharia Informática

Ano Letivo 2022/23

Inteligência Artificial

Entrega da Primeira Fase

Grupo 009

Ana Rita Poças (A97284) Inês Ferreira (A97372)
Joana Branco (A96584) João Braga (A97368)

30 de novembro de 2022

Índice

1 Introdução

Este relatório surge no âmbito da Unidade Curricular de Inteligência Artificial em que nos foi proposto o desenvolvimento de diversos algoritmos de procura para a resolução do jogo *VectorRace*.

O projeto encontra-se dividido em duas fases, e, nesta primeira fase, devemos ser capazes de ter pelo menos um circuito com um participante do jogo a encontrar o caminho mais curto, com o algoritmo de pesquisa que elegemos.

Ao longo deste relatório iremos descrever o problema que nos foi apresentado, a sua formulação, bem como os mecanismos que posssibilitaram o desenvolvimento da estratégia de procura que escolhemos.

2 Descrição do Problema

O *VectorRace* é um jogo de simulação de carros simplificado, que tem um conjunto de movimentos e regras que lhe estão associadas.

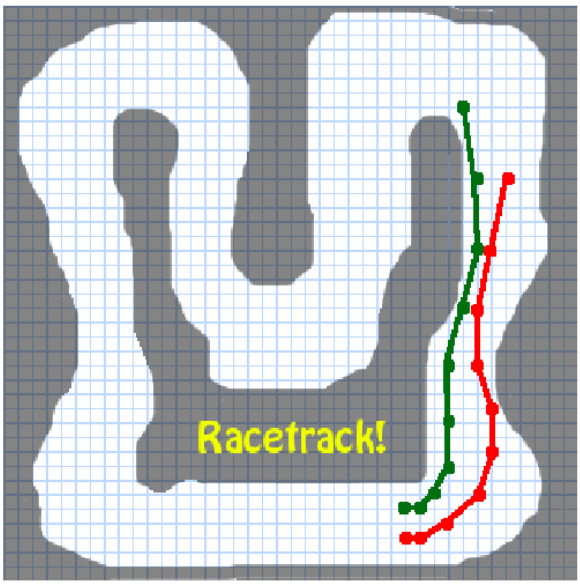

Figura 1: Exemplo de um circuito *VectorRace*

O circuito pode ser tomado como uma matriz que apresenta linhas e colunas, assim sendo num determinado instante o jogador estará numa posição p=(pl,pc). Desta forma, para se movimentar, num determinado instante, o carro pode acelerar -1,0 ou 1 unidades em cada direção, logo temos como acelerações possíveis -1,0,+1, com a=(al,ac) a representar a aceleração de um carro nas duas direções num determinado instante.

Sendo p=(pl,pc), o tuplo que representa a posição do jogador num determinado instante, para além da aceleração, o jogador deve ter associado a si um tuplo v=(vl,vc) que indica a velocidade do carro nessa jogada, que deve ser tido em consideração antes de efetuar a próxima jogada.

Sendo este jogo, uma simulação de carros existe a possibilidade de o carro poder sair dos limites da pista, e nesse caso o carro assume velocidade 0.

Cada movimento do carro numa jogada, de uma posição para outra, está associado a um custo de 1 unidade, à exceção do caso em que sai dos limites da pista, que terá o um custo de 25 unidades.

Um circuito no nosso problema terá a seguinte representação:

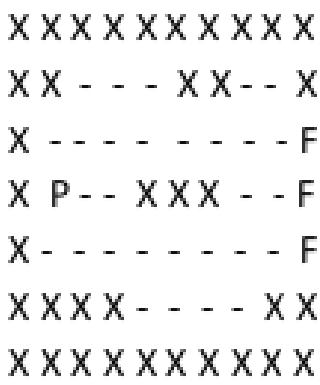

Figura 2: Representação de um circuito

Uma pista terá assim 4 elementos possívies:

- **"P"** que representa a posição inicial;
- **"F"** que representa a posições finais;
- **"X"** que representa um obstáculo/fora da pista;
- **" - "** que representa a pista;

3 Formulação do Problema

A formulação do problema tem por base a procura de uma sequência de ações que conduza a um estado desejável e, neste contexto, o objetivo do carro é atravessar a pista do ponto "P"até ao ponto "F"com o menor custo. O carro tem como possibilidades a movimentação do seu objeto no sentido vertical, horizontal e diagonal. De modo a representar todos os estados alternativos, decidiu-se criar um grafo. Consequentemente, estes podem ser analisados dando assim, uma sequência de ações que vai de encontro ao objetivo do problema. Pode-se concluir assim que este problema é do tipo de exploração, sendo que estamos perante um espaço de estados desconhecidos.

3.1 Representação do estado inicial

O estado inicial do problema consiste na representação do circuito da mesma forma que é apresentado no seu *input*. Este estado apresenta o circuito com a posição do jogador na sua localização inicial, sem haver qualquer tipo de alteração da mesma.

3.2 Estado Objetivo

O propósito do problema é encontrar um caminho com menor custo. Assim sendo, o estado objetivo deve ser representado pela posição do jogador na meta, após ter feito o seu percurso com uma estratégia de procura que resulte no menor custo.

3.3 Operadores

Os operadores baseiam-se nos movimentos possíveis para o jogador em causa. Neste caso, o jogador pode atravessar a pista com movimentos na horizontal, vertical e diagonal. Nesta primeira fase, o jogador poderá avançar apenas uma "casa", uma unidade da pista, de cada vez.

3.4 Custo da Solução

O custo da solução varia conforme a estratégia de procura e o caminho percorrido pelo jogador através da pista. Tendo em conta, a estratégia de procura não informada implementada neste problema, verificamos que o custo da solução é de 8 unidades, sendo esta a melhor solução.

4 Descrição das Tarefas Realizadas

O nosso programa corre com o seguinte comando ***python3 main.py*** e quando se executa o comando surge o seguinte menu que permite ao utilizador escolher a operação que pretende executar:

```
anarita@anarita-Nitro-AN515-55:~/IA$ python3 main.py
1 -> Gerar Circuito
2 -> Representar pista em forma de grafo
3 -> Obter o menor caminho - Problema de Custo Uniforme
Introduza a sua opção->
```

Figura 3: Menu do nosso programa

4.1 Circuito Gerado

A primeira operação disponível no nosso programa é a de Gerar um Circuito.

Nesta primeira fase do trabalho, de forma a gerarmos um circuito, criamos um ficheiro .txt do circuito e dentro da função **main** fazemos a abertura e leitura do ficheiro e guardamos o seu conteúdo dentro de um *array*. Depois, passamos esse *array* como argumento à função ***imprimeCircuito*** que está definida na classe **Matriz** no módulo **matrix.py**. Esta função percorre cada elemento do array e imprime-o no ecrã. A função ***imprimeCircuito*** é invocada na função **main** e, desta forma, quando damos como input a opção 1 é nos apresentado o seguinte circuito:

```
anarita@anarita-Nitro-AN515-55:~/IA$ python3 main.py
1 -> Gerar Circuito
2 -> Representar pista em forma de grafo
3 -> Obter o menor caminho - Problema de Custo Uniforme
Introduza a sua opção-> 1
X X X X X X X X X
X X - - - X X - - X
X - - - - - - - - F
X P - - X X X - - F
X - - - - - - - - F
X X X X - - - - X X
X X X X X X X X X X
```

Figura 4: Circuito gerado pelo nosso programa

4.2 Representação da pista em forma de grafo

De forma a representarmos a pista em forma de grafo, adotamos a seguinte estratégia:
Na classe **Matriz** do módulo **matrix.py**:

- Definimos uma função que transforma o *array* que guarda o circuito numa matriz, e, desta forma, temos os elementos presentes no circuito representados numa matriz. Uma vez tendo o circuito representado sobre forma de matriz, somos capazes de obter as coordenadas da posição inicial, isto é a posição onde temos um "P", através da função ***encontraPosicaoInicial***.
- De forma análoga, criamos, também, a função ***encontraPosicoesFinais*** que retorna uma lista com as posições da matriz onde temos um "F".
- Criamos, ainda, a função ***returnCustoOfaPos*** que recebe uma posição como argumento e retorna o seu custo, isto é, se nessa posição tiver um elemento "X"o custo será 25, caso contrário será 1.
- Depois, criamos a função ***getListofAdjencencies*** que retorna um dicionário em que temos cada posição da matriz associada à lista de posições que lhe são adjacentes.

Na classe **Arestas** do módulo **arestas.py**:

- Criamos a função ***addEdge*** que recebe o circuito, uma lista e 2 posições adjacentes (u e v), sendo que queremos ir de u para v. Esta função, adiciona à lista que recebe a aresta de u para v, e o custo da aresta, isto é o custo da posição v.
- Depois, criamos a função ***dictToGraph*** que, usando a função ***addEdge*** cria uma lista em que cada elemento tem a seguinte forma: (posição u do circuito, (posição v adjacente a u do circuito, custo da deslocação para a posição v))
- Criamos a função ***getEdges*** que, basicamente, pega na lista devolvida pela função ***dictToGraph*** e transforma numa outra lista com o formato (posição u do circuito, posição v adjacente a u do circuito, custo da deslocação para v).

Por último, na função ***main*** iteramos os elementos da presentes na lista devolvida por ***getEdges*** e imprimimos cada elemento no ecrã.

Figura 5: Excerto da representação da pista em forma de grafo pelo nosso programa

4.3 Estratégia de procura

Após termos estudado as estratégias de procura abordadas nas aulas, consideramos, nesta primeira fase, adotar uma **estratégia de procura não informada** e que a que melhor se adequava ao nosso programa era a **Estratégia de Procura de Custo Uniforme**.

Pelo facto de termos adotado uma estratégia de procura não informada, a implementação da velocidade e aceleração numa determinada posição, não são tidas em consideração.

Assim, para cada nó da lista de estados não expandidos, guardamos o custo total do caminho do estado inicial para esse nó e expandimos sempre o nó com menor custo da lista de estados não expandidos (medido pela função de custo da solução).

Definimos o algoritmo desta estratégia de procura no módulo **newgrafo.py**, através da definição das classes ***Nodo*** e ***oGrafo***.

A classe ***Nodo*** guarda os nodos da Heap e verifica se um dado nodo tem menor custo do que outro dado.

Para efeitos de simplificação, aquando da aplicação do algoritmo, ao invés de considerarmos um nodo da matriz (3,1), considerámo-lo como 31. Isto é ir de (3,1) para (3,2) é a mesma coisa no nosso programa que ir de 31 para 32.

A classe oGrafo recebe uma lista que consiste na transformação da ***getEdges*** que tem o formato ((3,1),(3,2),1) e transforma-a numa lista de inteiros do tipo (31,32,1), devolvendo assim a lista de arestas do grafo com o respetivo custo. A classe recebe, também, o número de nodos do grafo, isto é o número total de posições da matriz que representam os elementos do nosso circuito. Assim, com a função ***encontraCaminhoMaisCurto*** que recebe o grafo inicializado, a posição inicial do jogador e a lista de posições da meta e o número de nodos

do circuito, somos capazes de aplicar o algoritmo de procura de Custo Uniforme que devolve o caminho com o menor custo entre a posição inicial e a meta.

Assim, para o seguinte circuito:

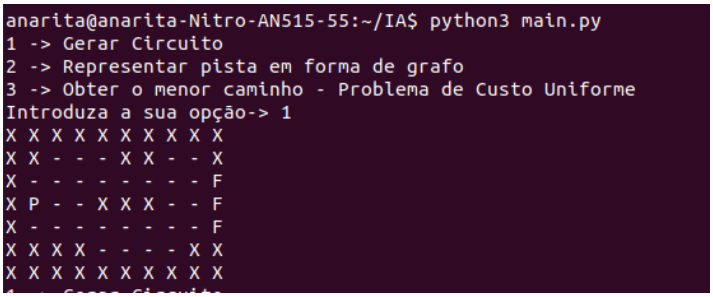

Figura 6: Circuito gerado pelo nosso programa

O caminho com menor custo devolvido pela nossa estratégia de procura é:

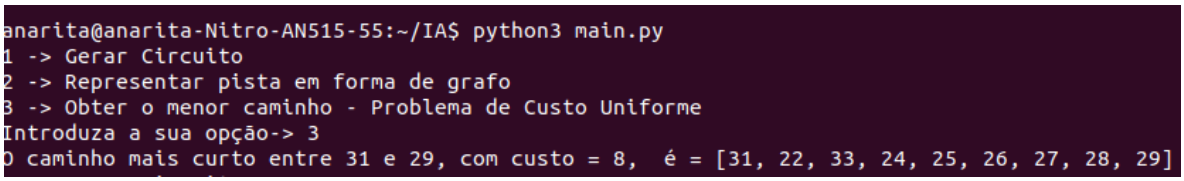

Figura 7: Caminho devolvido pelo programa

5 Sumário e discussão dos resultados obtidos

Por fim, e uma vez que fomos capazes de gerar um circuito *VectorRace*, representar a pista em forma de grafo e desenvolver uma estratégia de procura não informada, acreditamos que fomos capazes de atingir a grande maioria dos objetivos que nos foram propostos para esta primeira fase de desenvolvimento.

O caminho ótimo e o seu custo associado que obtivemos para o nosso circuito atual parece-nos bastante aceitável.

A estratégia de procura usada nesta fase, tem a vantagem de nos permitir, sempre, encontrar o caminho com o menor custo em cada estado. Outra vantagem é o facto de ser pouco complexa, quase linear. No entanto, pelo facto de termos de monitorizar os vértices que já foram visitados e o facto de estarmos a fazer uma Pesquisa Não Informada, o que faz com que consuma bastante tempo enquanto está em processamento, faz nos considerar que esta pode não ser a estratégia mais eficaz.

Desta forma, pretendemos na próxima fase implementar, também, pelo menos um algoritmo de pesquisa informada e, comparar os resultados obtidos com os desta fase.