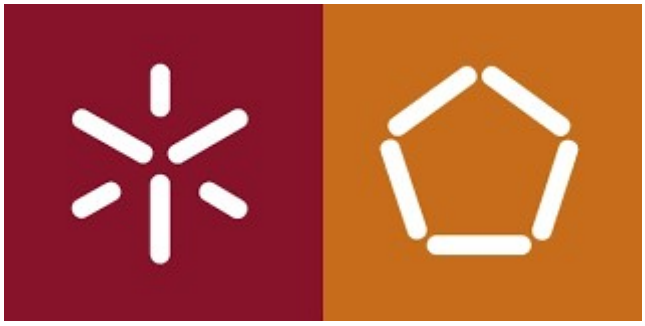

Universidade do Minho

Departamento de Informática

VectorRacer

Inteligência Artificial

Grupo 14

Afonso Magalhães- A95250

Marta Ribeiro- A95408

Artur Leite- A97027

Diana Teixeira- A97516

Índice

1. Introdução

Este trabalho foi realizado no âmbito da disciplina de Inteligência Artificial e consiste numa implementação de um conjunto de algoritmos de procura para um jogo conhecido como *VectorRace.* Trata-se de um jogo de simulação de corridas mais simplificado, que envolve movimentação vetorial dos veículos ao longo de uma pista. Neste trabalho foram utilizados diversos algoritmos de procura lecionados na disciplina, com o intuito de encontrar os caminhos ideias para os corredores chegaram à meta.

2. Elementos Necessários

No âmbito de planear as tarefas necessárias para um bom funcionamento do trabalho, optamos por enumerar um conjunto de conhecimentos e requisitos do programa, de forma a nos guiarmos ao longo da realização do mesmo. Após uma análise detalhada do enunciado, achamos os seguintes elementos os mais relevantes deste programa:

2.1. Circuitos

Para estabelecer um local de jogo, será necessário desenhar e gerar circuitos onde os nossos corredores irão utilizar as suas estratégias de procura para chegar à meta. Para tal, estes terão de ser navegáveis e deverá existir sempre um caminho possível até à meta, o que implica que teremos de garantir que a chegada não se encontra impedida por obstáculos. Eis um exemplo de um circuito desenhado por nós:

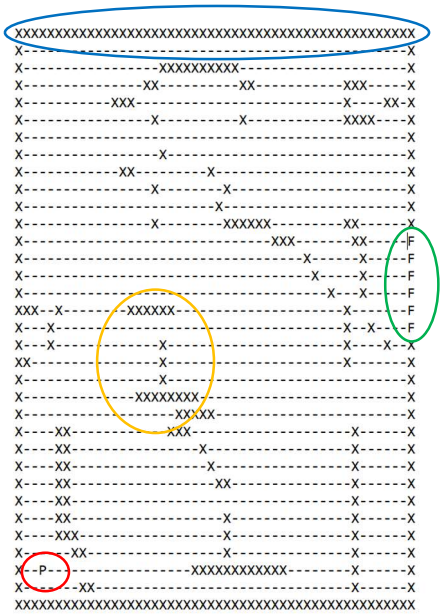

Figura 1- Exemplo de Circuito

- Ponto de Partida;
- Meta;
- Limites do Percurso;
- Obstáculos;

Como podemos ver na imagem, o mapa é perfeitamente navegável e a meta é acessível.

Com o intuito de fornecer a possibilidade de escolha de circuito na execução do programa, optamos por definir um conjunto de mapas pré-definidos, juntamente com um método capaz de gerar circuitos aleatórios.

2.2. Representação do Circuito em Grafo

Com o intuito de facilitar a implementação dos métodos de pesquisa, a representação do circuito num grafo é fulcral para otimizar os algoritmos que pretendemos utilizar. Para tal, representamos cada nodo como a posição atual do carro, juntamente com a sua velocidade respetiva. Depois, os seus nodos-filho vão representar as possíveis posições que ele pode assumir dependendo da sua velocidade atual e a sua decisão: se acelera, se trava, se muda de direção, etc. Tendo esta representação bem implementada, torna-se mais fácil implementar as estratégias de procura que visam chegar à meta com o menor custo possível.

2.3. Estratégias de Procura

Tal como mencionado acima, teremos de implementar um método de pesquisa pelo qual os nossos corredores irão seguir para atingir a meta. Para tal, optamos por implementar os métodos de pesquisa *Depth-First* (DF) e *Breadth-First* (BF).

- O método DF caracteriza-se por uma análise dos nodos todos num determinado ramo, voltando atrás quando não encontra o seu destino pretendido;
- O método BF caracteriza-se por uma análise de todos os nodos presentes num certo nível de profundidade, prosseguindo para o próximo nível assim que todos foram analisados;

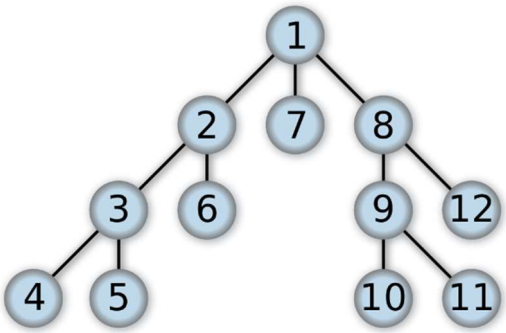

Figura 2- Pesquisa Depth-First, numerada por ordem de acesso

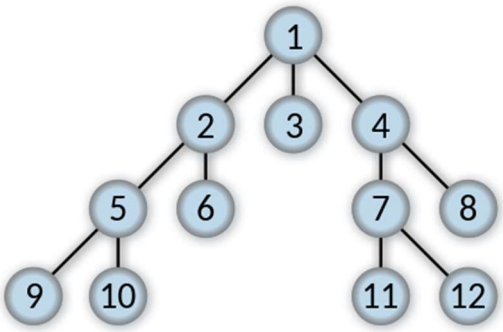

Figura 3- Pesquisa Breadth-First, numerada por ordem de acesso

2.4. Operadores e Parâmetros de decisão

Um dos principais fatores que influenciam as escolhas dos nossos pilotos é a velocidade em que se encontra o seu veículo. Através deste parâmetro, os pilotos irão decidir se em cada movimento irão acelerar o carro com o intuito de chegar à meta mais rápido, ou desacelerar com o intuito de evitar uma colisão com um obstáculo, evitando o custo pesado que um despiste atribui. Este parâmetro é caracterizado também pelas duas direções onde é considerado: linha e coluna.

Para além disso, tal como referi anteriormente, a aceleração vai ser um parâmetro de decisão fulcral na procura pela solução. A cada movimento feito pelo corredor, a sua escolha de aceleração é importante para garantir que este chega à meta com sucesso, visto que o valor que esta assume vai influenciar a velocidade, como vamos aferir na fórmula de cálculo da velocidade. Este parâmetro pode assumir três possíveis valores: {-1,0,1}.

Os valores da velocidade atual são dados pelo seguinte cálculo:

$$vl^{j+1} = vl^{j} + al$$

$$vc^{j+1} = vc^{j} + ac$$

Adicionalmente, a posição vetorial que o carro vai assumir dados os operadores e parâmetros que possui é obtida através dos seguintes cálculos

$$pl^{j+1} = pl^{j} + vl^{j} + al$$

$$pc^{j+1} = pc^{j} + vc^{j} + ac$$

Legenda:

$pl = posição\ na\ linha;\ pc = posição\ na\ coluna;\ vl = velocidade\ na\ linha$

$vc = velocidade\ na\ coluna;\ j = jogada;\ ac = aceleração\ na\ coluna;$

$al = aceleração\ na\ linha.$

3. Resultados de Execução

Para testar o nosso programa, utilizamos o seguinte mapa:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X-P-----XXXXX----------------X
X------------XXXXX--------XXXX
XXX---------------XXXX--XXX--X
X-X----X--------XXX----------X
X--X--X----XXXXX--XXXX-----XXX
X---XX-------------XXXXXX----X
X--XXXX-------------XXXXXXX--X
X--------XXXXXX---------X----F
X-----XXXXXXXXXX------------F
X---------X-----X------------F
X----XXXXX-----X-----XXXX----F
X------XXXXX------------XX---X
X------XX--------------X-----X
X------XX---X----------X-----X
X--------XXX--X-----XXX------X
X-------XXX----X----X--------X
X---------XX---X---X---------X
X--------XXXX-----XXX--------X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Figura 5- Mapa utilizado em formato .txt

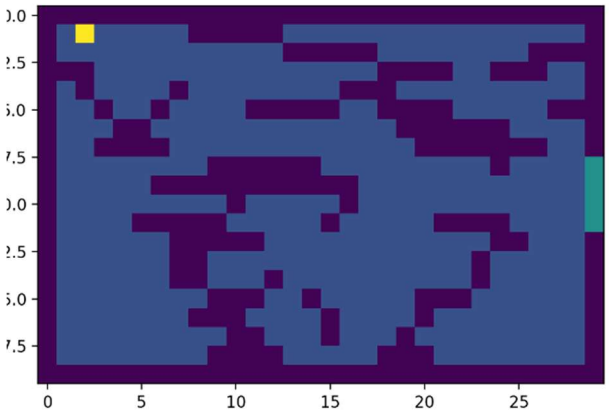

Figura 4- Mapa utilizado gerado em MatPlot

3.1. Algoritmo BF

O output do programa utilizando a estratégia BF, devolveu o seguinte conjunto de movimentos por parte do corredor: [‘(p (1, 2), v (0,0))’; ‘(p (1, 3), v (0,1))’; ‘(p (2, 5), v (1,2))’; ‘(p (4, 8), v (2,3))’; ‘(p (7, 12), v (3,4))’; ‘(p (9, 17), v (2,5))’; ‘(p (10, 23), v (1,6))’; ‘(p (10, 29), v (0,6))’]. Descreve aproximadamente o seguinte percurso:

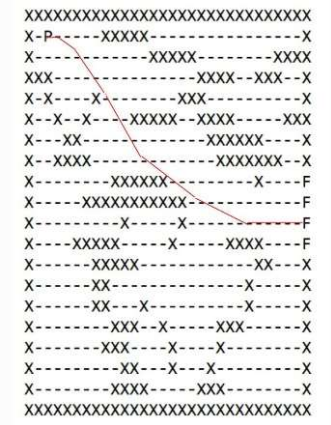

Figura 7- Representação do percurso BF no .txt

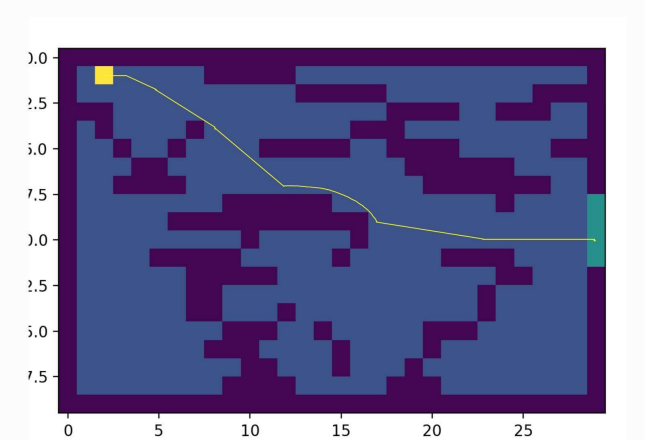

Figura 6- Representação do percurso BF no MatPlot

3.2. Algoritmo DF

O output do programa utilizando a estratégia BF, devolveu o seguinte conjunto de movimentos por parte do corredor: [‘(p(1, 2), v(0, 0))’; ‘(p(2, 3), v(1, 1))’; ‘(p(3, 3), v(1, 0))’; ‘(p(3, 4), v(0, 1))’; ‘(p(2, 6), v(-1, 2))’; ‘(p(1, 7), v(-1, 1))’; ‘(p(1, 7), v(0, 0))’; ‘(p(2, 8), v(1, 1))’; ‘(p(3, 9), v(1, 1))’; ‘(p(5, 9), v(2, 0))’; ‘(p(7, 8), v(2, -1))’; ‘(p(8, 6), v(1, -2))’; ‘(p(9, 4), v(1, -2))’; ‘(p(11,3), v(2, -1))’; ‘(p(12, 3), v(1, 0))’; ‘(p(13, 2), v(1, -1))’; ‘(p(13, 1), v(0, -1))’; ‘(p(12, 1), v(-1, 0))’; ‘(p(10, 2), v(-2, 1))’; ‘(p(8, 4), v(-2, 2))’; ‘(p(7, 7), v(-1, 3))’; ‘(p(7, 11), v(0, 4))’; ‘(p(8, 16), v(1, 5))’; ‘(p(10, 22), v(2, 6))’; ‘(p(11, 29), v(1, 7))’]. Descreve aproximadamente o seguinte percurso:

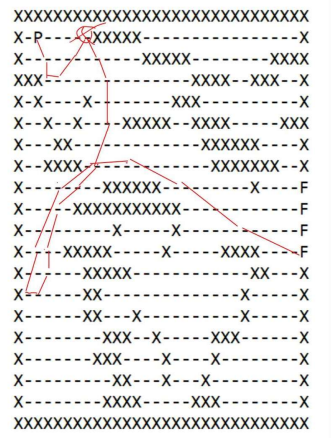

Figura 8- Representação do Percurso DF em .txt

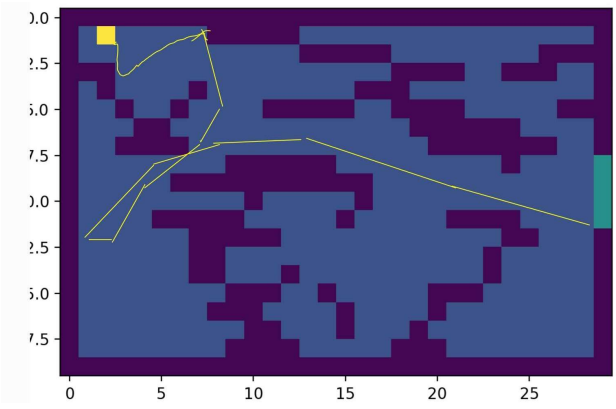

Figura 9- Representação do Percurso DF em MatPlot

3.3. Análise dos Resultados

Quanto ao resultado obtido na travessia *Breadth First,* surgiram dúvidas na nossa análise inicial, especificamente no 5º movimento onde o corredor vai da posição (7,12) para a posição (9,17).

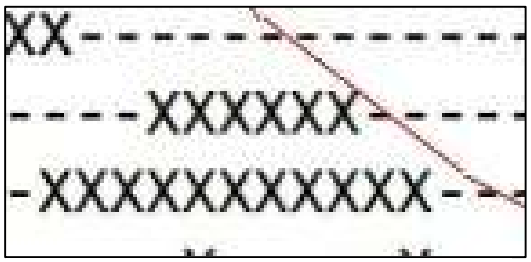

Figura 8- Excerto do mapa do 5º movimento

Analisando o excerto ao lado, podemos ver que o corredor passa rente aos obstáculos. Reavaliamos o nosso código para certificar se estes movimentos deviam ou não resultar numa colisão. Chegamos à conclusão que não é o caso, visto que na nossa implementação da expansão do grafo, o corredor verifica a direção do eixo cuja sua velocidade é maior primeiro, verificando posteriormente a outra direção. Tendo em conta que a sua velocidade na chegada é de (2,5), ele começou por verificar a sua linha, 3 espaços em frente, não encontrando um obstáculo. Verifica depois os 2 espaços na sua diagonal, não encontrado obstáculos também, validando assim este movimento.

Tomando em conta agora o resultado obtido na travessia *Depth First,* denotamos uma clara diferença em termos de desempenho comparativamente à travessia *Breadth First*, sendo notável o aumento de custo de um algoritmo para o outro. Nós chegamos à conclusão que isto se deve ao facto da DF não procurar os caminhos mais ótimos. Isto é, assim que a procura DF encontrar uma solução, essa é dada como resposta, ao invés de serem consideradas outras possibilidades.

4. Considerações do Tempo de Execução

Após a análise detalhada dos nossos resultados e da *performance* do sistema, nós denotamos que o nosso programa possuía um tempo de execução elevado, especificamente na geração do grafo que descreve os caminhos possíveis do condutor. Este problema era ainda mais preponderante quando o circuito em questão não possuía muitos obstáculos, que levava a um número elevado de possibilidades de movimento, resultando num grafo muito complexo.

Com o intuito de tentar analisar a possibilidade de o erro se realmente encontrar na geração do grafo, implementamos um conjunto de métodos que imprimiam os estados diferentes, juntamente com o número de nodos no grafo. Foi então que nos deparamos com um número de nodos muito inferior aos estados assumidos, o que nos levou à conclusão de que o carro estava a assumir estados que não devia.

5. Conclusão

Concluída esta primeira fase do projeto, acreditamos que fomos capazes de implementar um bom comportamento do nosso sistema, nomeadamente na consideração dos parâmetros que influenciam os movimentos do condutor. Apesar disso, reconhecemos que o movimento dos carros não está otimizado, pretendendo melhorar essa vertente na segunda fase.