
Inteligência Artificial

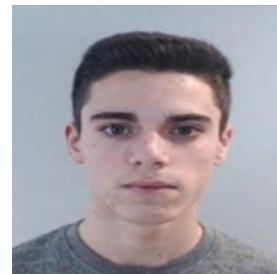
TRABALHO REALIZADO POR:

XAVIER SANTOS MOTA
TELMO JOSÉ PEREIRA MACIEL
PEDRO DANTAS DA CUNHA PEREIRA
DANIEL JOSÉ SILVA FURTADO

Grupo 58



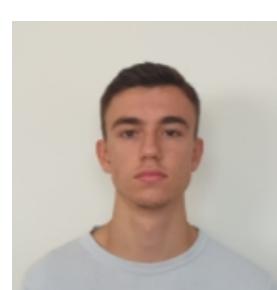
A88220
Xavier Mota



A96569
Telmo Maciel



A97396
Pedro Pereira



A97327
Daniel Furtado

Índice

1	Introdução	2
2	Contextualização do Problema	3
3	Formulação do Problema	4
3.1	Nodo	4
3.2	Estado Inicial	4
3.3	Estado Objetivo	4
3.4	Operadores	4
3.4.1	Mantém X e mantém Y	4
3.4.2	Mantém-se X e aumenta Y	4
3.4.3	Mantém-se X e reduz Y	4
3.4.4	Aumenta X e mantém Y	5
3.4.5	Aumenta X e aumenta Y	5
3.4.6	Aumenta X e reduz Y	5
3.4.7	Reduz X e mantém Y	5
3.4.8	Reduz X e aumenta Y	5
3.4.9	Reduz X e reduz Y	5
3.5	Custo da Solução	5
4	Estratégias Usadas	6
4.1	Representação do Circuito como Grafo	6
4.2	Implementação do Ambiente Competitivo	6
4.2.1	Posições Iniciais e Ponto "P"	6
4.2.2	Carros na Mesma Posição	7
4.3	Implementação de Velocidades	7
4.4	Algoritmos de Procura	7
4.5	Resultados Obtidos	8
4.5.1	Resultados dos Diferentes Algoritmos	9
4.5.2	Comparação de Resultados	10
5	Circuitos	10
6	Menu	11
6.1	Escolha de Circuito	11
6.2	Número de Jogadores	11
6.3	Menu Geral	12
6.4	Opções de Impressão	12
6.4.1	Opção 1 - Imprimir Mapa	12
6.4.2	Opção 2 - Imprimir Grafo	13
6.4.3	Opção 3 - Imprimir Nodos do Grafo	13
6.4.4	Opção 4 - Imprimir Arestas do Grafo	14
6.5	Modos de Travessia	14
7	Conclusão	16

1 Introdução

No âmbito da unidade curricular de Inteligência Artificial, foi-nos proposto a realização de um trabalho prático que consiste em desenvolver algoritmos de procura para a resolução de um jogo, nomeadamente o VectorRace, também conhecido como RaceTrack.

O VectorRace é um jogo de simulação de carros simplificado, que contém um conjunto de movimentos e regras associados.

Nesta segunda fase do trabalho foi-nos pedida a incorporação de diferentes algoritmos de procura, tanto informada como não informada, no nosso trabalho. Na última fase foram implementados os algoritmos BFS (*Breath-First Search*) e DFS (*Deep-First Search*), ambos não informados, pelo que desta vez procurámos desenvolver os algoritmos AStar (A^*) e Gulosa (*Greedy*), sendo estes dois algoritmos de procura informada. Para além disto, foi também proposto o desenvolvimento de um ambiente capaz de suportar, pelo menos, dois participantes.

Por fim, é também necessária a implementação de acelerações e velocidades seguindo as especificações fornecidas no enunciado.

Posto isto, quaisquer alterações à formulação do problema serão apresentadas e justificadas, bem como os resultados obtidos aquando da utilização das diferentes estratégias de procura.

2 Contextualização do Problema

Como já foi referido na introdução, o VectorRace é um jogo de simulação de carros simplificado, que contém um conjunto de movimentos e regras associados.

Os diferentes circuitos utilizados são apresentados em forma de matriz, de modo a possibilitar a representação das diferentes posições nas quais os carros se poderão encontrar ao longo das pistas em formato de coordenadas, no qual as linhas serão representadas por \mathbf{y} e as colunas por \mathbf{x} .

Cada circuito contém secções nas quais o carro se pode movimentar, podendo estas ser vistas como partes da própria pista, e secções nas quais isto não será possível, podendo estas ser consideradas fora dos limites da pista ou obstáculos da mesma.

O carro poderá ser capaz de 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 $a = (a_x, a_y)$ 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 $j(p^j = ((p_x, p_y))$, e v o tuplo que indica a velocidade do carro nessa jogada $(v^j = ((v_x, v_y))$, na seguinte jogada o carro irá estar na posição:

- $p_y^{j+1} = p_y^j + v_y^j + a_y$
- $p_x^{j+1} = p_x^j + v_x^j + a_x$

A velocidade do carro num determinado instante é calculada:

- $v_y^{j+1} = v_y^j + a_y$
- $v_x^{j+1} = v_x^j + a_x$

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 ou bater numa parede, o custo é de 25 unidades.

3 Formulação do Problema

Tivemos que proceder a algumas alterações na formulação do problema em relação à fase anterior uma vez que implementamos a velocidade.

3.1 Nodo

Nesta fase fizemos uma alteração na classe *Nodo*, uma vez que nesta fase aplicamos as velocidades. Neste caso a alteração que efetuamos foi que ao invés de termos um tuplo só com as coordenadas passamos a ter um tuplo com as coordenadas e velocidades do tipo (coordenada X, coordenada Y, velocidade X, velocidade Y).

3.2 Estado Inicial

No estado inicial do jogo vamos ter dois jogadores, jogador **A** e jogador **B**, que representa a posição inicial dos mesmos. Estas posições são dadas por *input* do utilizador ao longo do programa.

3.3 Estado Objetivo

No estado objetivo do jogo, os carros encontrar-se-ão ambos na posição final do circuito, representada por **F** no mapa.

3.4 Operadores

3.4.1 Mantém X e mantém Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro mantém-se constante quer na componente X quer na componente Y

3.4.2 Mantém-se X e aumenta Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro mantém-se constante na componente X e aumenta na componente Y

3.4.3 Mantém-se X e reduz Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro mantém-se constante na componente X e diminui na componente Y

3.4.4 Aumenta X e mantém Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro aumenta na componente X e mantém-se constante na componente Y

3.4.5 Aumenta X e aumenta Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro aumenta na componente X e aumenta também na componente Y

3.4.6 Aumenta X e reduz Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro aumenta na componente X e reduz na componente Y

3.4.7 Reduz X e mantém Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro reduz na componente X e mantém-se constante na componente Y

3.4.8 Reduz X e aumenta Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro reduz na componente X e aumenta na componente Y

3.4.9 Reduz X e reduz Y

Pré-Condição: Não existe muro na posição para a qual o carro se vai movimentar

Efeito: A velocidade do carro reduz na componente X e reduz também na componente Y

```
acel = [(0,1), (0,-1), (1,0), (1,1), (1,-1), (-1,1), (-1,0), (0,0), (-1,-1)]
```

Figure 1: Acelerações possíveis

3.5 Custo da Solução

Como foi referido na contextualização do problema, o custo da solução será calculado com base em cada movimento realizado pelo carro, sendo que, por cada alteração na velocidade acima indicada, o custo final aumenta em 1 unidade. Adicionalmente, sempre que o carro entrar numa zona do mapa inválida, o custo final terá um aumento de 25 unidades.

4 Estratégias Usadas

Nesta secção do relatório serão expostas e explicadas as diferentes estratégias utilizadas para a implementação das diversas funcionalidades necessárias para a realização do trabalho, bem como a apresentação e discussão dos resultados obtidos após a finalização do projeto.

4.1 Representação do Circuito como Grafo

Nesta fase, para conseguir implementar as velocidades no "jogo" tivemos que alterar a forma como criávamos o nosso grafo e utilizamos a seguinte estratégia:

Partindo da posição "P", a posição pela qual começamos a gerar o grafo, assinalada no circuito ainda em forma de matriz, o nosso grafo vai-se expandindo para as posições adjacentes válidas considerando também a velocidade, isto é, o grafo partindo de uma posição inicial vai gerando todas as outras posições para onde o jogador se pode deslocar aplicando as condições de verificação de posições válidas, não podendo ser uma "parede", e também as condições de velocidade e aceleração.

A construção do grafo termina quando todas as possibilidades que o jogador pode tomar forem já verificadas e validadas. No final, ao analisarmos o grafo, será possível verificar que existem vários nodos com a mesma posição mas com velocidades diferentes visto que um jogador pode alcançar a mesma posição mas com diferentes velocidades dependendo de vários fatores.

Desta forma, alteramos a forma de representação dos nodos do grafo. Cada nodo é caracterizado na forma de um tuplo com quatro parâmetros: os dois primeiros para a posição e os outros dois para a velocidade.

Ainda no nosso trabalho são calculadas as heurísticas, que devido à introdução das velocidades e com a implementação da procura informada são necessárias. No nosso trabalho decidimos que a heurística de cada nosso corresponde à distância euclidiana entre o respetivo nodo e a meta da pista, neste caso representada por "F".

4.2 Implementação do Ambiente Competitivo

4.2.1 Posições Iniciais e Ponto "P"

Dada a necessidade da implementação de um ambiente competitivo capaz de suportar 2 jogadores simultaneamente, e após uma discussão interna do grupo em relação ao tema, foi decidido que não faria sentido ambos os jogadores começarem no mesmo ponto de partida, anteriormente representado por P, sendo que isto seria visto como os carros estarem "amontoados". Desta forma, aquando da inicialização do jogo, cada jogador poderá escolher as coordenadas iniciais do seu carro.

Apesar disto, e como foi explicado anteriormente, o ponto P continuará presente no mapa, uma vez que é a partir deste ponto que começamos a criação do grafo que representa o circuito.

4.2.2 Carros na Mesma Posição

No enunciado do trabalho foi-nos dada a opção de escolher a maneira que consideramos mais adequada para resolver as seguintes condições:

- Caso, no ambiente competitivo, exista possibilidade de dois participantes se dirigirem para a mesma célula da pista, apresentar e justificar a decisão tomada nessas condições

Posto isto, o grupo decidiu, mais uma vez, que não faria sentido os carros se encontrarem na mesma posição em qualquer momento do jogo. De modo a seguir esta linha de pensamento, optámos por alterar as coordenadas de um dos carros sempre que ambos se encontrassem na mesma posição, movimentando-o para a coluna anterior e tornando assim as suas posições adjacentes.

Visto que as coordenadas de um dos carros, nestas condições, terá de ser alterada, verificámos se a futura posição é ou não válida, de modo a não prejudicar nenhum dos jogadores. Caso isto não se verifique, em vez de movimentarmos o carro para a coluna anterior, este será movido para a linha anterior. Se ainda assim a posição for inválida, este é movimentado para a linha e coluna anteriores.

4.3 Implementação de Velocidades

Tendo como objetivo a implementação de velocidades nesta fase do trabalho o nosso grupo decidiu que a melhor forma de implementar as velocidades era diretamente no grafo quando é criado. Enquanto o grafo está a ser gerado cada nodo já possui a informação da velocidade que o jogador terá caso passe pelo respetivo nodo. Isto quer dizer que qualquer algoritmo estará sempre sobre a influência do fator velocidade. Aplicando todas as acelerações possíveis à velocidade que o nodo "pai" possui e verificando quais são as posições válidas. Também é importante referir que na nossa perspetiva o jogador pode ter simultaneamente velocidade na coordenada dos X's e na coordenada dos Y's. Assim o jogador poderá movimentar-se na diagonal.

4.4 Algoritmos de Procura

Como nos foi pedido para esta fase do projeto, para além dos algoritmos de procura não informada (BFS e DFS) que implementamos anteriormente, adicionámos também ao programa os algoritmos *Greedy* e *AStar*, sendo que estes são ambos algoritmos de procura informada.

- ***Greedy*:** O algoritmo *Greedy* é um método de procura, bastante eficiente, que tenta encontrar a solução ótima num problema tomando sempre a decisão que parece ser a mais promissora naquele momento, sem se preocupar com o impacto dessa escolha no futuro, o que pode levar a escolhas subótimas em alguns casos.

O algoritmo segue iterativamente, escolhendo sempre a opção com a melhor pontuação até chegar ou à solução ótima ou a um ponto onde não há mais opções viáveis.

- **AStar:** O algoritmo A* é um algoritmo de procura heurística utilizado para encontrar o caminho mais curto entre dois pontos num grafo. O algoritmo expande os nós do grafo em ordem crescente de um custo estimado, que é composto pela distância já percorrida até o nó atual mais a distância estimada do nó atual até o destino. Isso permite que o algoritmo A* encontre o caminho mais curto possível com uma grande eficiência.

4.5 Resultados Obtidos

Para testar a implementação dos 4 algoritmos de procura usou-se um mapa como exemplo e testou-se com um único jogador, sempre a partir da mesma posição, neste caso coordenadas (1,1).

```

X X X X X X X X X X X X X X X X
X - - - - - P - - - - - X
X - - - - - - - - - - - X
X - - X X X - - X X X - - X
X - - X X X - - X X X - - X
X - - - X X - - X X - - - X
X - - - X X - - X X - - - X
X - - - X X X X - - - X
X - - - - X X - - - - X
X - - X X - X X - X X - - X
X - - - X X X X X X - - - X
X - - - - - - - - - - - X
X - - - X X X X X X X X - - X
X - - - X X X X X X X X - - X
X - - - X X X X X X X X - - X
X - - - X X X X X X X X - - X
X - - - X X X X X X X X - - X
X - - - X X X X X X X X - - X
X - - - - - - - - - - - X
X - - - - - X X - - - - - X
X - - - - - X X - - - - - X
X - - - - - - - - - - - X
X X X X X X X F X X X X X X X

```

Escolha uma opção de corrida para o Jogador0: 1
 Escolha a coordenada inicial válida X do Jogador0 entre 1 e 12: 1
 Escolha a coordenada inicial válida Y do Jogador0 entre 1 e 22: 1

 Coordenadas iniciais do Jogador1: (1, 1)

Figure 2: Mapa escolhido para testar os algoritmos

4.5.1 Resultados dos Diferentes Algoritmos

Nesta secção vamos apresentar o resultado da aplicação dos diferentes algoritmos, isto é, o caminho que o jogador percorre e o custo da solução.

Podemos ver na figura abaixo a implementação do algoritmo DFS, que teve como solução um custo de 213.

```
Resultados da DFS:
[('1, 1, 0, 0'), ('1, 2, 0, 1'), ('1, 4, 0, 2'), ('1, 7, 0, 3'), ('1, 11, 0, 4'), ('1, 16, 0, 5'), ('1, 20, 0, 4'), ('1, 23, 0, 3'), ('1, 20, 0, 0'), ('1, 21, 0, 1'), ('1, 23, 0, 2'), ('1, 21, 0, 0'), ('1, 22, 0, 1'), ('1, 22, 0, 0'), ('1, 21, 0, -1'), ('1, 19, 0, -2'), ('1, 18, 0, -1'), ('1, 18, 0, 0'), ('1, 19, 0, 1'), ('1, 21, 0, 2'), ('2, 22, 1, 1'), ('3, 22, 1, 0'), ('4, 23, 1, 1'), ('3, 22, 0, 0'), ('3, 21, 0, -1'), ('3, 21, 0, 0'), ('3, 22, 0, 1'), ('4, 22, 1, 0'), ('5, 23, 1, 1'), ('4, 22, 0, 0'), ('4, 21, 0, -1'), ('4, 21, 0, 0'), ('4, 22, 0, 1'), ('5, 22, 1, 0'), ('6, 23, 1, 1'), ('5, 22, 0, 0'), ('5, 21, 0, -1'), ('6, 22, 1, 0'), ('7, 21, 1, -1'), ('6, 22, 0, 0'), ('7, 22, 1, 0'), ('8, 23, 1, 1'), ('7, 22, 0, 0'), ('7, 23, 0, 1')]
Com um custo total de : [213]
```

Figure 3: Algoritmo DFS

De seguida apresentamos a execução do algoritmo BFS, que teve um custo de 46.

```
Resultados da BFS:
[('1, 1, 0, 0'), ('2, 1, 1, 0'), ('2, 2, 0, 1'), ('2, 4, 0, 2'), ('1, 7, -1, 3'), ('1, 11, 0, 4'), ('1, 16, 0, 5'), ('0, 20, -1, 4'), ('1, 16, 0, 0'), ('1, 17, 0, 1'), ('1, 17, 0, 0'), ('2, 18, 1, 1'), ('2, 19, 0, 1'), ('2, 21, 0, 2'), ('1, 22, -1, 1'), ('1, 22, 0, 0'), ('2, 22, 1, 0'), ('3, 22, 1, 0'), ('4, 22, 1, 0'), ('5, 22, 1, 0'), ('6, 22, 1, 0'), ('7, 22, 1, 0'), ('7, 23, 0, 1')]
Com um custo total de : [46]
```

Figure 4: Algoritmo BFS

Agora temos o algoritmo GREEDY com um custo de 37.

```
Resultados da GREEDY:
[('1, 1, 0, 0'), ('2, 1, 0, 1'), ('2, 2, 0, 2'), ('2, 4, 0, 1'), ('2, 7, 0, 3'), ('1, 11, -1, 4'), ('1, 16, 0, 5'), ('2, 20, 1, 4'), ('4, 23, 2, 3'), ('2, 20, 0, 0'), ('3, 21, 1, 1'), ('5, 22, 2, 1'), ('6, 22, 1, 0'), ('7, 22, 1, 0'), ('7, 23, 0, 1')]
Com um custo total de : [37]
```

Figure 5: Algoritmo GREEDY

Por último, o algoritmo ASTAR, com um custo solução de 12.

```
Resultados da ASTAR:
[('1, 1, 0, 0'), ('2, 2, 1, 1'), ('2, 4, 0, 2'), ('2, 7, 0, 3'), ('1, 11, -1, 4'), ('1, 14, 0, 3'), ('2, 17, 1, 3'), ('3, 19, 1, 2'), ('4, 21, 1, 2'), ('5, 22, 1, 1'), ('6, 22, 1, 0'), ('7, 22, 1, 0'), ('7, 23, 0, 1')]
Com um custo total de : [12]
```

Figure 6: Algoritmo ASTAR

4.5.2 Comparação de Resultados

Como podemos observar na secção anterior, todos os algoritmos têm um custo solução diferente, o que quer dizer que cada algoritmo possui uma eficiência diferente. Como era de esperar os algoritmos de procura não informada tem um custo maior em relação aos algoritmos de procura informada e, como se repara, o algoritmo ASTAR é muito mais eficiente que os restantes possuindo um custo de 12 (comparado com os restantes custos de 37, 46 e 213).

5 Circuitos

Tal como na fase anterior do trabalho, e como se pôde verificar ao longo do relatório, foram criados diferentes circuitos, de forma não aleatória, de modo a possibilitar o bom funcionamento do programa. Os mapas disponibilizados para esta fase foram alterados e são os que apresentamos abaixo:

```
x x x x x x x x x x x x x x x x x x x x x x x x x
x - x x x x - x x x x x x x - - x x x
x x x x x x x - - x x x x x x - - x x
x - - - - - x x - - x - - x - -
x - - x x x x x - x - x x x x x
x - - - - - x x - - x - - x - -
x x x x - - x x x - - x x x - - x x
x - x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
x - x x x x x x x x x x x x x x x x x x x
```

Figure 7: deserto.txt

```
x x x x x x x x x x x x x x x x x x x x x x x x x
x - - - - - - x - - - - x - - - - x - - - - x
x - x - - x - - x - - x - - x - - x - - x x x
x x - - x - - x - - x - - x - - x - - x - - x - F
x - - x - - x - - x - - x - - x - - x - - x - - x
x x - - x - - x - - x - - x - - x - - x - - x - x
x - - x - - x - - x - - x - - x - - x - - x - x - x
x P - x - - - - x - - - - x - - - - x - - - - x
x - - x - - x - - x - - x - - x - - x - - x - - x
x - x - - x - - x - - x - - x - - x - - x - - x
x - x - - x - - x - - x - - x - - x - - x - - x
x - x - - x - - x - - x - - x - - x - - x - - x
```

Figure 8: praia.txt

```
x x x x x x x x x x x x x x x x x x x
x x x - P - - - x x x
x x - - - - - - - x x
x - - - - x - - - - x
x - - - x x x x - - x
x - - x x x x x x - - x
x - - x x x x x x - - x
x - - x x x x x x - - x
x - - x x x x x x - - x
x - - x x x x x x - - x
x - - - x - - - - x
x - - - - - - - - x
x x x x x x F x x x x x
```

Figure 9: deserto.txt

```
x x x x x x x x x x x x x x x x x x x
x - - - - - P - - - - x
x - - - - - - - - - x
x - x x x - - x x x - - x
x - - x x x - - x x x - - x
x - x x - - x x - - x
x - - x x x x x - - x
x - - - x x x x - - x
x - - - x x x x x x - - x
x - - x x x x x x x - - x
x - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - x x x x x x x - - x
x - - - - - - - - - x
x - - - - x x - - - - x
x - - - - - - - - - x
x x x x x x x x F x x x x x x x
```

Figure 10: praia.txt

6 Menu

6.1 Escolha de Circuito

Assim que o programa é executado é-nos apresentado o seguinte menu com as diferentes opções para circuitos que disponibilizámos, sendo que para escolher um basta fornecer o respetivo número como *input*:



Figure 11: Seleção do Circuito

6.2 Número de Jogadores

Após a seleção de um mapa é nos dada a opção de jogar com 1 ou 2 jogadores sendo que, novamente, devemos fornecer como *input* o respetivo número da opção que escolhermos:

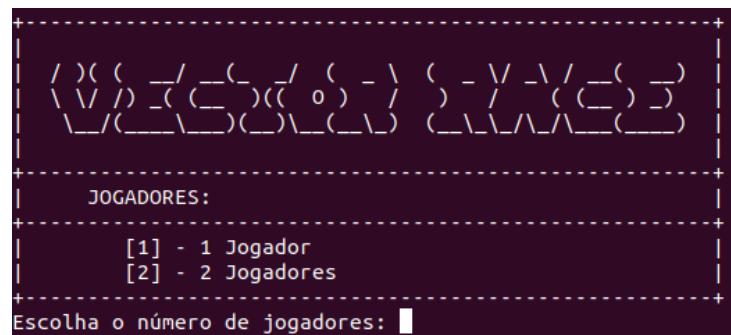


Figure 12: Seleção do Número de Jogadores

6.3 Menu Geral

Depois de escolhermos o número de jogadores chegamos finalmente ao menu geral do programa, no qual nos são apresentadas as diferentes funcionalidades e opções para o que é possível fazer no jogo:

```
***** MENU *****
* [1] - Imprimir Mapa *
* [2] - Imprimir Grafo *
* [3] - Imprimir Nodos do Grafo *
* [4] - Imprimir Arestas do Grafo *
* [5] - Modos de Travessia *
* [0] - Sair *
*****
Escolha uma opção: █
```

Figure 13: Menu Geral

6.4 Opções de Impressão

Como podemos verificar na figura anteriormente apresentada, as quatro primeiras opções do menu geral dizem respeito a impressões. Estas serão seguidamente explicadas e expostas separadamente.

6.4.1 Opção 1 - Imprimir Mapa

Caso esta opção seja escolhida, o mapa será impresso da mesma forma que se encontra representado no ficheiro de texto:

```
***** MENU *****
* [1] - Imprimir Mapa *
* [2] - Imprimir Grafo *
* [3] - Imprimir Nodos do Grafo *
* [4] - Imprimir Arestas do Grafo *
* [5] - Modos de Travessia *
* [0] - Sair *
*****
Escolha uma opção: 1

X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
X - - - - - - X - - - - X - - - - X - - - - X - - - - X
X - X - - X - - X - - X - - X - - X - - X - - X - X X X
X X - - X - - X - - X - - X - - X - - X - - X - - X - F
X - - X - - X - - X - - X - - X - - X - - X - - X - - X
X X - - X - - X - - - - X - - X - - X - - X - - X - - X
X P - X - - - - X - - - - X - - - - X - - - - X - - - - X
X - X - - X - - X - - X - - X - - X - - X - - X - - X X
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
Pressione qualquer tecla para avançar. █
```

Figure 14: Imprimir Mapa

6.4.2 Opção 2 - Imprimir Grafo

Caso esta opção seja escolhida, serão apresentados todos os nodos pertencentes ao grafo, assim como todas as suas ligações aos nodos vizinhos:

```
***** MENU *****
*
* [1] - Imprimir Mapa *
* [2] - Imprimir Grafo *
* [3] - Imprimir Nodos do Grafo *
* [4] - Imprimir Arestas do Grafo *
* [5] - Modos de Travessia *
* [0] - Sair *
*
*****
Escolha uma opção: 2

Node (1, 6, 0, 0): [((1, 7, 0, 1)', 1), ((1, 5, 0, -1)', 1), ((2, 6, 1, 0)', 1), ((2, 7, 1, 1)', 1), ((0, 7, -1, 1)', 1), ((0, 6, -1, 0)', 1)]
Node (1, 7, 0, 0): [((1, 7, 0, 0)', 0), ((2, 8, 1, 1)', 1), ((2, 7, 1, 0)', 1), ((0, 8, -1, 1)', 1), ((1, 8, 0, 1)', 1)]
Node (1, 5, 0, -1): [((1, 6, 0, 0)', 1)]
Node (2, 6, 1, 0): [((4, 5, 2, -1)', 2), ((2, 7, 0, 1)', 1), ((2, 6, 0, 0)', 0), ((3, 6, 1, 0)', 1)]
Node (2, 7, 1, 1): [((1, 6, 0, 0)', 1)]
Node (0, 7, -1, 1): [((1, 6, 0, 0)', 1)]
Node (1, 7, 0, 0): [((1, 8, 0, 1)', 1), ((1, 6, 0, -1)', 1), ((2, 7, 1, 0)', 1), ((2, 8, 1, 1)', 1), ((0, 8, -1, 1)', 1), ((0, 7, -1, 0)', 1)]
Node (2, 8, 1, 1): [((1, 7, 0, 0)', 1)]
Node (2, 7, 1, 0): [((1, 7, 0, 0)', 1)]
```

Figure 15: Imprimir Grafo

6.4.3 Opção 3 - Imprimir Nodos do Grafo

Caso esta opção seja escolhida, serão apresentados todos os nodos do grafo:

```
***** MENU *****
*
* [1] - Imprimir Mapa *
* [2] - Imprimir Grafo *
* [3] - Imprimir Nodos do Grafo *
* [4] - Imprimir Arestas do Grafo *
* [5] - Modos de Travessia *
* [0] - Sair *
*
*****
Escolha uma opção: 3

[(1, 6, 0, 0), (1, 7, 0, 1), (1, 5, 0, -1), (2, 6, 1, 0), (2, 7, 1, 1), (0, 7, 0),
(2, 2, 0, -1), (2, 0, 0, -3), (3, 0, 1, -3), (3, 6, 0, 1), (3, 5, 0, 0), (4,
-1, 2), (2, 3, 0, 0), (3, 4, 0, -1), (2, 5, -1, 0), (2, 5, 0, 1), (1, 5, -1,
(3, 4, 1, 1), (0, 6, -2, 1), (0, 5, -2, 0), (2, 7, 0, 2), (2, 4, 1, 0), (5,
1, 0), (3, 2, 0, 0), (3, 0, 0, -2), (4, 1, 1, -1), (4, 2, 1, 0), (4, 0, 1, -2
(4, 1, 0, 0), (4, 0, 0, -1), (5, 0, 1, -1), (4, 3, 0, 1), (4, 2, 0, 0), (5, 2,
```

Figure 16: Imprimir Nodos do Grafo

6.4.4 Opção 4 - Imprimir Arestras do Grafo

Caso esta opção seja escolhida, todas as arestas do grafo serão apresentadas, sendo que serão exibidos os nodos inicial e final de cada aresta, bem como os seus custos respetivos:

```
*****  
*          MENU          *  
*****  
*  
*  [1] - Imprimir Mapa      *  
*  [2] - Imprimir Grafo      *  
*  [3] - Imprimir Nodos do Grafo  *  
*  [4] - Imprimir Arestras do Grafo  *  
*  [5] - Modos de Travessia      *  
*  [0] - Sair                  *  
*  
*****  
Escolha uma opção: 4  
  
(1, 6, 0, 0) ->(1, 7, 0, 1) custo:1  
(1, 6, 0, 0) ->(1, 5, 0, -1) custo:1  
(1, 6, 0, 0) ->(2, 6, 1, 0) custo:1  
(1, 6, 0, 0) ->(2, 7, 1, 1) custo:1  
(1, 6, 0, 0) ->(0, 7, -1, 1) custo:1  
(1, 6, 0, 0) ->(0, 6, -1, 0) custo:1  
(1, 7, 0, 1) ->(1, 7, 0, 0) custo:0  
(1, 7, 0, 1) ->(2, 8, 1, 1) custo:1  
(1, 7, 0, 1) ->(2, 7, 1, 0) custo:1  
(1, 7, 0, 1) ->(0, 8, -1, 1) custo:1
```

Figure 17: Imprimir Arestras do Grafo

6.5 Modos de Travessia

Caso a opção do menu geral escolhida seja a opção 5 (Modos de Travessia), é-nos apresentado um novo menu, sendo que este será utilizado para inicializar propriamente o jogo. Inicialmente são fornecidas as diferentes opções de travessia para o jogador, tendo este que selecionar uma delas.

```
*****  
*          TRAVESSIA          *  
*****  
*  
*  [1] - DFS                  *  
*  [2] - BFS                  *  
*  [3] - GREEDY                *  
*  [4] - ASTAR                 *  
*  
*****  
  
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  
X - - - - - X - - - - X - - - - X - - - - X - - - - X  
X - X - - X - - X - - X - - X - - X - - X - - X - X X  
X X - - X - - X - - X - - X - - X - - X - - X - - X - F  
X - - X - - X - - X - - X - - X - - X - - X - - X - - X  
X X - - X - - X - - - - X - - X - - X - - X - - X - - X  
X P - X - - - - X - - - - X - - - - X - - - - X - - - - X  
X - X - - X - - X - - X - - X - - X - - X - - X - - X X  
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  
  
Escolha uma opção de corrida para o Jogador: ■
```

Figure 18: Modos de Travessia

Depois disto são pedidas as coordenadas iniciais para o jogador e, após estas serem selecionadas, é apresentado o resultado do algoritmo de procura selecionado, bem como o custo da solução. Seguidamente é apresentada a simulação da corrida.

```

X X X X X X X X X X X X X X X X
X - - - - - P - - - - - X
X - - - - - - - - - - X
X - - X X X - - X X X - - X
X - - X X X - - X X X - - X
X - - X X - - X X - - X
X - - X X X X X X - - X
X - - X X X X X X - - X
X - - X X X X X X - - X
X - - X X X X X X - - X
X - - X X X X X X - - X
X - - X X X X X X - - X
X - - - - - - - - - - X
X - - - - - - - - - - X
X - - - - - - - - - - X
X - - - - - - - - - - X
X X X X X X X F X X X X X X

```

Escolha uma opção de corrida para o Jogador 0: 1
 Escolha a coordenada inicial válida do Jogador0 entre 1 e 12; 1
 escolha a coordenada inicial válida y do jogador0 entre 1 e 22; 1

 Coordenadas iniciais do Jogador0: (1, 1)

Resultados da DFS:
 ['(1, 1, 0, 0)', '(1, 2, 0, 1)', '(1, 4, 0, 2)', '(1, 7, 0, 3)', '(1, 11, 0, 4)', '(1, 16, 0, 5)', '(1, 20, 0, 4)', '(1, 23, 0, 3)', '(1, 20, 0, 0)', '(1, 21, 0, 1)', '(1, 23, 0, 2)', '(1, 21, 0, 0)', '(1, 22, 0, 0)', '(1, 22, 0, 1)', '(1, 21, 0, 2)', '(1, 19, 0, -1)', '(1, 18, 0, -2)', '(1, 18, 0, -1)', '(1, 21, 0, 1)', '(1, 21, 0, 2)', '(2, 22, 1, 1)', '(3, 22, 1, 0)', '(4, 23, 1, 1)', '(5, 22, 1, 0)', '(4, 21, 0, -1)', '(4, 22, 0, 0)', '(4, 21, 0, 1)', '(5, 22, 1, 0)', '(6, 23, 1, 1)', '(5, 22, 0, 0)', '(5, 21, 0, -1)', '(5, 21, 0, 0)', '(6, 22, 0, 0)', '(7, 21, 1, 0)', '(6, 22, 0, 0)', '(7, 22, 1, 0)', '(8, 23, 1, 1)', '(7, 22, 0, 0)', '(7, 23, 0, 1)']
 Com um custo total de : [213]

Pressione alguma tecla para avançar para a simulação da corrida: [

Figure 19: Exemplo Resultados

De notar que, caso sejam 2 jogadores, o primeiro seleciona o método de travessia e as coordenadas iniciais e só depois poderá o segundo jogador fornecer os seus *inputs*.

No final é possível observar a simulação iterativa dos jogadores ao percorrerem o mapa com o algoritmo correspondente, oferecendo uma melhor percepção como o algoritmo funciona e também é uma funcionalidade que melhora bastante a interação com o jogo.

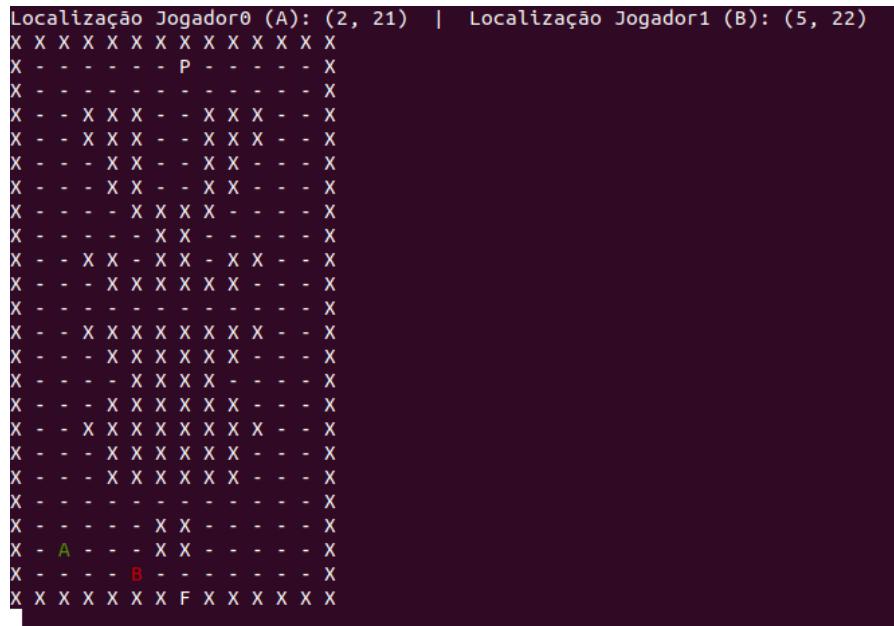


Figure 20: Tracking de 2 jogadores em simultâneo

7 Conclusão

Após o término deste trabalho prático de grupo da Unidade Curricular de Inteligência Artificial, o grupo considera, mais uma vez, que a realização desta fase do projeto foi bastante satisfatória e bem sucedida, tendo em conta que alcançámos todos os objetivos estipulados para o projeto dentro do prazo definido.

Com o desenrolar do projeto deparamo-nos com algumas dificuldades, que fomos ultrapassando, com maior ou menor facilidade.

Consideramos que este projeto foi importante para obter uma melhor compreensão do funcionamento e das diferenças de resultados dos diferentes algoritmos de procura, tanto informada como não informada.

Assim, consideramos que foi alcançada uma melhor consolidação da matéria lecionada nas aulas, permitindo, ainda, a exploração de novos domínios necessários à concretização do nosso projeto.