Projeto Inteligência Artificial 2021/2022

Ricardo Franco
2202314
2202314@my.ipleiria.pt

Guilherme Fernandes
2200666
2200666@my.ipleiria.pt

ABSTRACT
Neste relatório, são apresentadas todas as características e funcionalidades do projeto desenvolvido baseado no jogo "Mummy Maze", e é feita uma análise relativa aos resultados obtidos.

1. INTRODUÇÃO
Baseado no jogo "Mummy Maze", este projeto tem como objetivo desenvolver um sistema que consiga encontrar um conjunto de ações executadas pelo herói de forma a que o mesmo consiga chegar à saída de cada nível vivo. Para ser possível alcançar este objetivo foram utilizados algoritmos de procura, dotados de inteligência artificial.

2. DESCRIÇÃO DA SOLUÇÃO
Uma solução é um conjunto de estados em que cada estado é um sucessor ao estado anterior, e o estado final corresponde a um estado em que o herói está vivo e está na célula mais próxima da célula da saída.

3. DESCRIÇÃO DE PROBLEMA
Sendo que uma solução é um conjunto de estados, o problema é caracterizado como sendo do tipo Best-First Problem.

Neste problema as entidades jogam por turno, em que a primeira entidade a jogar é sempre o herói e subsequentemente os inimigos existentes no estado atual. O herói tem de se deslocar da sua posição inicial até à célula mais próxima da saída contornando as paredes e portas, sendo que não pode ser morto por nenhum inimigo, ou passar por cima de nenhuma armadilha. As portas serão abertas ou fechadas sempre que uma entidade, herói ou inimigos, passarem por cima da chave.

Podem existir níveis onde haja mais que um inimigo, logo será necessário que os inimigos consigam eliminar outros inimigos para que o nível tenha solução. Caso o herói consiga fazer com que dois inimigos ocupem a mesma célula, elas lutaram entre si, restando apenas a múmia que se deslocou. A múmia eliminada desaparecerá do nível.

4. DESCRIÇÃO DO ESTADO
Um estado é representado através de uma matriz 13x13, sendo que graficamente a matriz é apresentada como sendo 6x6. Um estado contém uma célula que armazena a posição da saída, uma célula com a posição de onde o herói devia estar para ganhar o jogo, uma lista com a posição de cada porta existente, uma lista com as posições de cada armadilha e uma lista com todos os inimigos do nível. É composto também por uma célula com a posição da chave caso exista e uma uma variável com as informações do herói.

4.1 Representação dos elementos do jogo

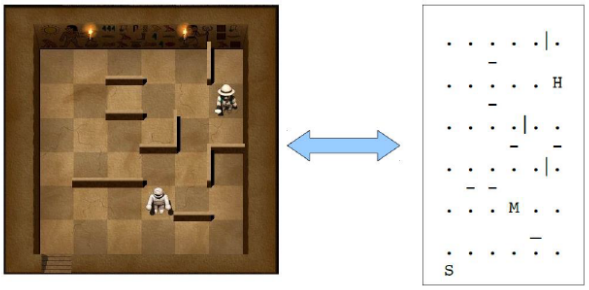

Figura 1. ***Representação gráfica do problema e representação matricial.***

Para as várias representações dos elementos do jogo foram criados vários caracteres para a representação de cada um, dos quais:

'|', para uma posição ocupada por uma parede vertical; (VERTICAL_WALL)

'–', para uma posição ocupada por uma parede horizontal; (HORIZONTAL_WALL)

'H', para o agente; (HERO)

'.', para uma posição vazia; (EMPTY)

'S', para a localização da saída; (EXIT)

M', para a múmia branca; (WHITEMUMMY)

'V', para a múmia vermelha; (REDMUMMY)

'A', para a armadilha; (TRAP)

'E', para o escorpião; (SCORPION)

'C', para a chave; (KEY)

'=', para porta horizontal fechada; (HORIZONTAL_CLOSE)

'_', para porta horizontal aberta; (HORIZONTAL_OPEN)

'"', para porta vertical fechada; (VERTICAL_CLOSE)

")", para porta vertical aberta. (VERTICAL_OPEN)

5. ELEMENTOS DO PROBLEMA
Cada entidade desloca-se de forma diferente sendo que os inimigos deslocam-se sempre de forma a se aproximarem do herói, de forma a matá-lo.

5.1 Herói

O herói consegue deslocar-se apenas uma casa de cada vez em cada turno. Isto é: em cada turno o herói pode passar para a quadrícula imediatamente acima, abaixo, à esquerda ou à direita da quadrícula onde se encontra, a não ser que haja algum obstáculo, como uma parede, a bloquear esse caminho. O herói pode ainda optar por não se movimentar durante o seu turno.

5.2 Múmia Branca

A múmia branca desloca-se até duas casas em cada turno. O objetivo da múmia é matar o herói e para isso vai tentando deslocar-se para a posição onde este está. A múmia branca primeiro procura estar na mesma coluna onde se encontra o herói e só depois é que tenta deslocar-se para a mesma linha que este.

5.3 Múmia Vermelha

Tal como a múmia branca, a múmia vermelha pode realizar até dois movimentos em cada turno. A múmia vermelha procura primeiro estar na mesma linha que o herói e só depois é que se desloca para a mesma coluna.

5.4 Escorpião

O escorpião desloca-se da mesma forma que a múmia branca, embora apenas se desloque uma quadrícula por turno.

5.5 Chave

Sempre que uma entidade, herói ou inimigo, passa pela quadrícula onde se encontra a chave, as portas existentes no estado serão abertas ou fechadas. Se a porta estiver fechada passa a estar aberta e vice-versa.

5.6 Parede

Uma parede bloqueia os movimentos do herói e seus inimigos, sendo que a única opção que as entidades podem tomar é contornar a mesma.

5.7 Porta

Quando está fechada, a porta funciona como uma parede: bloqueia os movimentos do herói e seus inimigos.

5.8 Armadilha

O herói morre sempre que pisar uma armadilha. As armadilhas não têm efeito nos inimigos.

6. HEURÍSTICAS

As funções heurísticas são usadas pelos algoritmos de procura informados e têm como objetivo avaliar um estado de forma a estimar o custo de ir do estado atual até a um estado objetivo. O valor de uma heurística é igual a 0 quando o estado avaliado é o estado objetivo, é igual ou superior a 0 quando o estado atual não é o estado objetivo ou assume um valor infinito quando o estado objetivo é impossível de ser alcançado através do estado atual.

6.1 Tile distance

Nesta heurística pretende-se dar uma melhor avaliação heurística ao estado em que o herói estiver mais perto da célula da saída.

6.2 Number of enemy possible moves relative

Nesta heurística pretendemos obter o somatório de movimentos possíveis de cada inimigo no estado atual relativamente ao tipo de inimigo. Sendo que como a múmia branca e o escorpião tentam primeiro andar para a mesma coluna do herói só é somado o número de movimentos que esses inimigos podem dar horizontalmente, para a direita ou esquerda. Como a múmia vermelha tenta primeiro deslocar-se para a mesma linha que o herói, só é somado o número de movimentos que a múmia vermelha pode dar verticalmente, acima ou abaixo.

6.3 Number of enemy possible moves

Nesta heurística pretende-se obter o número possível de movimentos do inimigo, um estado é quanto melhor quanto menor for o número de movimentos que um inimigo possa fazer.

6.4 Number of hero possible moves

Nesta heurística pretende-se obter o número possível de movimentos do herói, um estado é quanto melhor quanto maior for o número de movimentos possíveis do herói.

6.5 Number of enemies

Nesta heurística pretende-se obter o estado em que tenha o menor número de inimigos possível. Um estado é melhor quanto menor o número de inimigos

6.6 Enemies wall in direction exit

Nesta heurística pretende-se obter o número de inimigos que tem uma parede na direção da saída, um estado é melhor quanto maior o número de inimigos com parede em direção à saída.

6.7 Distance between hero and enemies

Nesta heurística pretende-se obter a distância entre o inimigo e o herói, um estado é melhor quanto maior for a distância entre o herói e o inimigo.

6.8 Distance between enemies

Nesta heurística pretende-se obter a distância entre inimigos, um estado é melhor quanto menor for a distância entre inimigos, pois a probabilidade de se eliminarem aumenta.

7. ANÁLISE DE ESTATÍSTICAS

Neste capítulo vamos estudar os resultados obtidos ao resolver todos os níveis fornecidos pelos professores, ao analisar gráficos criados através dos dados gerados por alguns algoritmos de procura e refletir sobre o desempenho das heurísticas implementadas.

Após a realização dos testes para verificar se o programa conseguia resolver os níveis fornecidos, inferimos que todos os níveis têm solução sendo que no nível 21_v1 é encontrada solução pela razão evidenciada no ponto 8.1 deste relatório.

7.1 Algoritmos não informados

Para o estudo dos algoritmos não informados vamos analisar alguns níveis e comparar o desempenho entre o algoritmo "Breadth First Search" e o algoritmo "Deep First Search".

7.1.1 Níveis onde foram gerados mais nós utilizando o algoritmo "Breadth First Search"

Ao analisarmos os resultados obtidos utilizando o algoritmo "Breadth First Search", verificamos que existem 3 níveis em que o número de nós gerados é superior aos restantes. Sendo esses os níveis 7, 11, 17.

Ao serem gerados mais nós conseguimos perceber que para estes níveis o herói possui um maior número de movimentos possíveis por estado.

Tabela 1. Resultados obtidos do nível 7 usando o algoritmo Breadth First Search

Solution cost	19.0
Num of expanded nodes	415
Max frontier size	44
Num of generated nodes	1181
Duration	0.003

Tabela 2. Resultados obtidos do nível 11 usando o algoritmo Breadth First Search

Solution cost	15.0
Num of expanded nodes	594
Max frontier size	75
Num of generated nodes	1576
Duration	0.014

Tabela 3. Resultados obtidos do nível 11 usando o algoritmo Breadth First Search

Solution cost	17.0
Num of expanded nodes	612
Max frontier size	96
Num of generated nodes	1723

7.1.2 Nível 4 utilizando "Depth First Search" e "Limited Depth Search"

Durante o estudo dos resultados reparamos que ao utilizarmos os algoritmos "Depth First Search" e o "Limited Depth Search", sendo o "Depth First Search" usado pelo "Limited Depth Search", no nível 4 o número de nós gerados alcançava o valor de 130768, e com isso conseguimos perceber que o algoritmo fez uma pesquisa em profundidade expandindo vários nós que não levam à solução durante bastantes iterações.

Tabela 4. Resultados obtidos do nível 4 usando o algoritmo Depth First Search

Solution cost	40.0
Num of expanded nodes	64111
Max frontier size	37
Num of generated nodes	130768

Duration	0.676

7.1.3 Comparação entre o desempenho dos algoritmos "Breadth First Search" e o "Depth First Search" no nível 12

Depois de estudarmos o desempenho dos dois algoritmos ao resolverem o nível 12 reparamos que o custo da solução encontrada pelo algoritmo "Breadth First Search" é inferior ao custo da solução encontrada pelo algoritmo "Depth First Search". Com isso percebemos que pelo facto do algoritmo "Depth First Search" não ser um algoritmo de procura ótimo, a solução encontrada poderia não ser a solução com o menor custo, e que ao utilizarmos o algoritmo "Breadth First Search" era possível descobrir a solução ótima, visto que este algoritmo já é caracterizado como ótimo.

Tabela 5. Resultados obtidos do nível 12 usando o algoritmo Breadth First Search

Solution cost	13.0
Num of expanded nodes	245
Max frontier size	30
Num of generated nodes	634

Tabela 6. Resultados obtidos do nível 12 usando o algoritmo Depth First Search

Solution cost	15.0
Num of expanded nodes	307
Max frontier size	20
Num of generated nodes	631

7.1.4 Comparação do número de nós gerados entre os algoritmos "Breadth First Search" e "Depth First Search" em todos os níveis

Para uma melhor visualização das estatísticas geradas pelos algoritmos, foram criados gráficos de colunas.

Ao analisar estes gráficos foi possível aferir que utilizando o algoritmo "Depth First Search" os níveis onde foram gerados mais nós foram o nível 21 e as versões desse mesmo nível, sendo essa observação diferente utilizando o algoritmo "Breadth First Search".

Podemos verificar também que o número máximo de nós gerados quando o algoritmo "Breadth First Search" é utilizado é maior do que quando o algoritmo "Depth First Search" é usado.

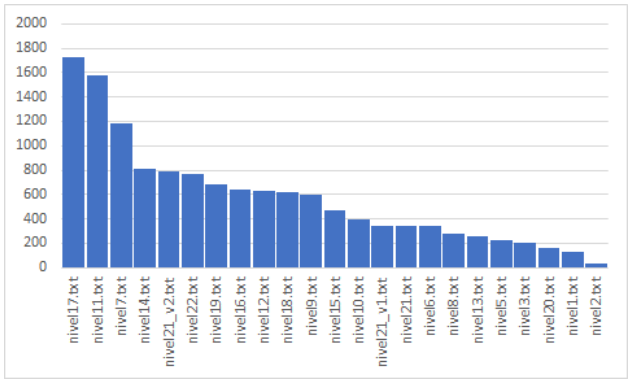

Figura 2. Gráfico com o número de nós gerados por nível ao utilizar o algoritmo Breadth First Search

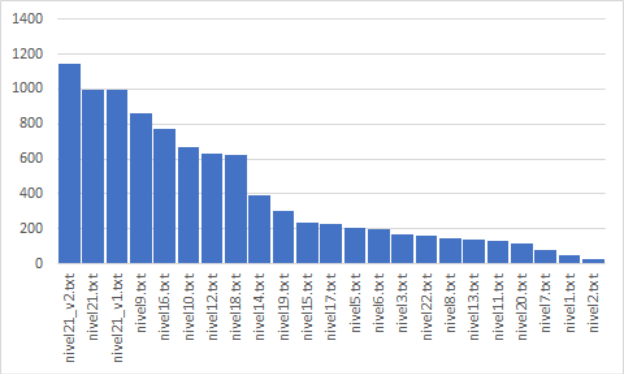

Figura 3. Gráfico com o número de nós gerados por nível ao utilizar o algoritmo Depth First Search

7.2 Algoritmos Informados

Para o estudo dos algoritmos informados vamos analisar algumas heurísticas implementadas com o intuito de comparar o desempenho das mesmas por nível usando o algoritmo de procura informado "A* Search".

7.2.1 Tiles distance to exit

Após a recolha de estatísticas relativas a esta heurística é possível destacar que em 88% dos níveis foi uma heurística admissível e que nos restantes 12% dos níveis subestimou o custo para chegar à solução.

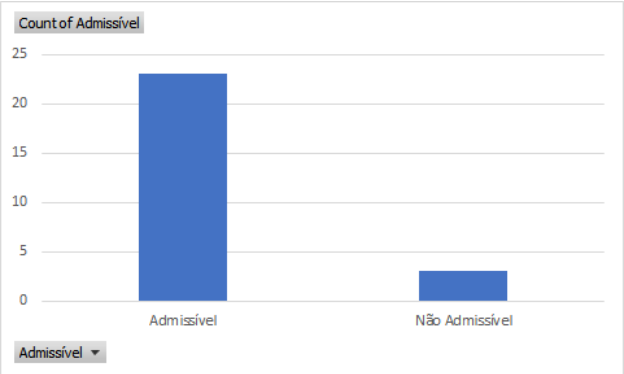

Figura 4. Gráfico com o número de níveis em que a heurística não foi e foi admissível.

7.2.2 Desempenho geral das heurísticas implementadas

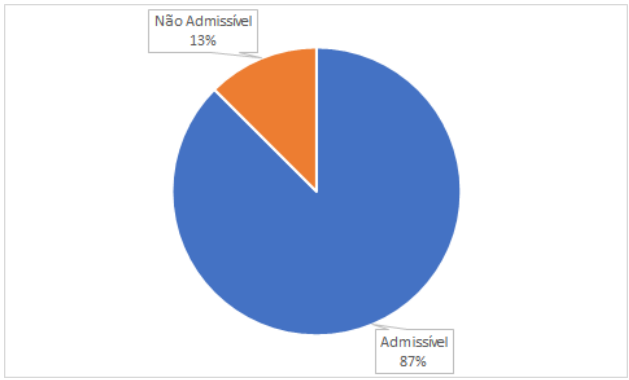

Figura 5. Gráfico Circular com a percentagem onde todas as heurísticas foram, ou não foram admissíveis.

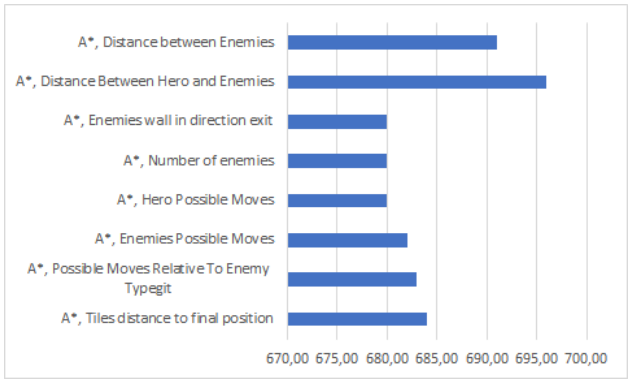

Figura 6. Gráfico de barras horizontais com somatório do custo das soluções encontradas por heurística.

7.2.2.1 Heurísticas com melhor desempenho

As heurísticas com um melhor desempenho foram as "Hero possible moves", "Number of enemies" e "Enemies wall in direction exit", visto que cada uma só não foi admissível num nível e o somatório do custo das soluções encontradas é o menor, 680.

7.2.2.2 Nível onde um maior número de heurísticas falharam

No nível 21_v2 foi onde existiu um maior número de heurísticas a subestimar o valor do custo real até à solução, cerca de 75%.

7.2.2.3 Heurística com pior desempenho

A heurística com pior desempenho foi a "Number of enemy possible moves relative", onde não foi admissível em cerca de 4 níveis.

7.3 Diferença entre algoritmos informados e não informados

Devido ao facto dos algoritmos informados terem um melhor conhecimento do problema, no gráfico apresentado abaixo é observado que existe uma grande diferença entre o número de nós gerados pelos dois tipos de algoritmos.

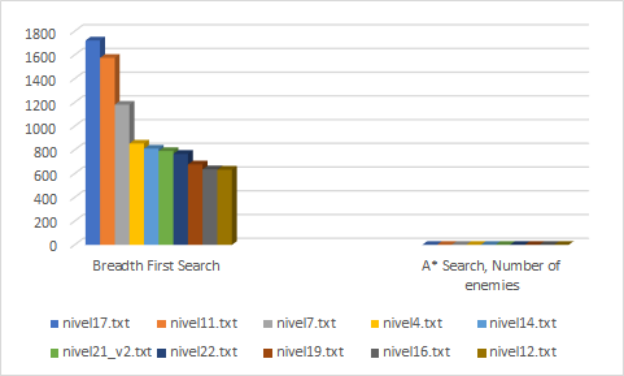

Figura 7. Gráfico com o número de nós gerados pelos algoritmos "Breadth First Search" e "A* Search", utilizando a heurística "Number of enemies", nos níveis apresentados.

8. EXTRAS

8.1 Nivel21_v1

Após termos jogado este nível podemos reparar que o nosso nível consegue obter solução devido à ordem em que colocamos os inimigos na lista dos inimigos, ou seja se o primeiro inimigo a mexer fosse o que estivesse mais perto do herói o nível não iria ter solução.

8.2 Inimigos passar por cima da chave

No enunciado é nos dito "Sempre que o Herói passa pela quadrícula onde se encontra a chave, abre ou fecha uma porta", mas tendo como base o jogo original, podemos reparar que os inimigos também podem abrir e fechar a porta passando por cima da chave, e tomamos partido dessa informação e implementamos esse extra no nosso projeto.

8.3 IDA*

Foi implementado um algoritmo de procura que não tinha sido implementado nas aulas práticas com o nome de "Iterative Deepening A star search". Tem semelhanças com o algoritmo "iterative deepening depth-first search", mas em vez de impor limites à profundidade de busca, impõe limites sucessivos ao custo das soluções que podem ser calculadas.

8.4 Iterative Deepening Search

Ao desenvolvermos o gerador de estatísticas verificamos que algumas vezes a utilização do algoritmo de procura "Iterative Deepening Search" era demorada. Para conseguirmos ultrapassar esse problema implementamos um tempo limite de execução ao mesmo, de forma a que passado 1 segundo de execução a procura seria interrompida.

8.5 Gerar estatisticas

Para conseguirmos fazer uma análise mais detalhada aos resultados obtidos de cada algoritmo após a execução do programa, foi implementado um gerador de estatísticas em que para cada estatística criada são criados ficheiros Excel com as informações desejadas.

8.6 Verificação se a heurística é admissível

Para conseguirmos verificar se uma heurística é admissível adicionamos uma funcionalidade ao gestor de estatísticas onde depois de se resolver o problema usando o algoritmo "Breadth First Search", sendo este um algoritmo não informado ótimo, guardamos o custo da solução encontrada por este algoritmo. Depois quando o algoritmo "A* Search" for usado no gerador de estatísticas, sendo este um algoritmo informado que usa as heurísticas e que é ótimo e completo só e só se a heurística for admissível, verificamos se o custo da solução encontrada pelo algoritmo "A* Search" é diferente do custo da solução ótima encontrada pelo algoritmo "Breadth First Search". Se sim então a heurística não é admissível.

9. CONCLUSÃO

Através da realização e desenvolvimento deste trabalho foi-nos possível implementar um programa que resolvesse o problema descrito, sendo que ao implementar a solução, adicionando alguns extras, a mesma ficou mais completa.

Por conseguinte, com a evolução deste programa foi-nos possível aprofundar mais o conhecimento sobre cada algoritmo de pesquisa que foi usado, relacionando, sempre, com a matéria lecionada, na cadeira de Inteligência Artificial, ao longo do semestre, o que se tornou muito importante para ambos os alunos, uma vez que nos permitiu estudar e conhecer, mais aprofundadamente a matéria relacionada com os algoritmos de pesquisa.

Por fim, este foi um trabalho muito importante, com base supra referido, sendo que não nos foi possível encontrar nenhuma heurística que fosse admissível para todos os níveis.