Relatório do 1º Trabalho Prático de Inteligência Artificial

Henrique Raposo nº33101   José Serra nº 33289

27 Março, 2019

1 Introdução

Neste primeiro trabalho temos como objetivo a exploração dos seguintes tópicos:

1. Pesquisa Não Informada
2. Pesquisa informada / Heuristicas

O exercício proposto, é o de pesquisar um caminho, entre dois pontos, segundo uma das pesquisas anteriores, numa matriz de tamanho NxN, neste caso N = 30.

2 Resolucão de problemas como problemas de pesquisa no espaço de estados

2.1 Estados e Operações:

Decidimos em primeiro lugar, representar os estados, como posições X e Y numa matriz NxN.

```
%estado_inicial((agente, sala inicial))
estado_inicial((18,18)).

%estado_final((agente, sala final)).
estado_final((26,26)).
```

De seguida aplicámos as restrições das casas ás quais o agente não pode aceder, e definimos as operações necessárias para o agente se poder mover pela matriz.

```
%bloqueadas(casa_inicial, casa_final)
bloqueadas((1,2), (1,3)).
bloqueadas((1,3), (1,2)).
bloqueadas((2,3), (2,2)).
bloqueadas((2,2), (2,3)).
bloqueadas((3,4), (4,4)).
bloqueadas((4,4), (3,4)).
bloqueadas((4,5), (3,5)).
bloqueadas((3,5), (4,5)).


%OPERACOES PARA QUE O AGENTE SE POSSA MOVER NA MATRIZ
```

```prolog
%op(Estado_atual, operador, estado_seguinte, custo)
%DESCER
op((X, Y), desce, (Z, Y), 1) :-
                profundidade(Prof),
                X < Prof,
                Z is X+1,
                ( visited((Z,Y))
                        -> fail
                        ; ( bloqueadas((X, Y), (Z, Y))
                                -> fail
                                ; asserta(visited((Z,Y)))
                          )
                ).

%DIREITA
op((X, Y), dir, (X, Z), 1) :-
                largura(Larg),
                Z is Y+1,
                Y < Larg,
                ( visited((X,Z))
                        -> fail
                          ;(bloqueadas((X, Y), (X, Z))
                                  -> fail
                                  ; asserta(visited((X,Z)))
                          )
                ).

%SUBIR
op((X, Y), sobe, (Z, Y), 1) :-
                X > 1,
                Z is X-1,
                ( visited((Z,Y))
                        -> fail
                          ; ( bloqueadas((X, Y), (Z, Y))
                                -> fail
                                ; asserta(visited((Z,Y)))
                          )
                ).

%ESQUERDA
op((X, Y), esq, (X, Z), 1) :-
                Y > 1,
                Z is Y-1,
                ( visited((X,Z))
                        -> fail
                         ; ( bloqueadas((X, Y), (X, Z))
                                 -> fail
                                 ; asserta(visited((X,Z)))
                           )
                 ).
```

Ao corrermos os algoritmos detectámos erros causados pelo facto de não termos controlo sobre os nós visitados o que resultou em loops infinitos, como tal, utilizámos o predicado dynamic do prolog de maneira a guardar o caminho por onde o agente vai passando.

```prolog
:- dynamic(visited/1).
    visited((18, 18)). %estado_inicial
```

3 Ex. 1

3.1 Alinea a)

Ao compararmos a solução dos vários algoritmos enunciados, chegámos à conclusão que o mais eficiente para a resolução deste problema é a pesquisa em Profundidade, isto porque comparativamente aos demais algoritmos, nomeadamente as pesquisas em largura e iterativa, apresenta um número inferior de nós visitados.
No entanto, temos em consideração que o número de nós visitados pela pesquisa em Profundidade poderia ser superior, caso a nossa ordem de operações a executar fosse diferente.

3.2 Alinea b)

Algoritmos de Pesquisa	Largura	Profundidade	Iterativa
nós visitados	472	87	2217
Custo	16	86	16
Tempo PC Henrique	0,037s	0,023s	0.105s
Tempo PC José	0,044s	0,045s	0.088s
número máximo nós em memória	50	93	30

A Partir dos dados da tabela anterior é possível concluir que o algoritmo de pesquisa em largura é o melhor para encontrar a solução do exercício enunciado, isto porque em comparação com os demais algoritmos, apresenta um custo igual à pesquisa em profundidade Iterativa, no entanto o seu desempenho temporal é superior em ambas as máquinas.

4 Ex. 2

4.1 Alinea a)

Relativamente ás heurísticas, decidimos utilizar duas, faladas pela docente nas aulas, nomeadamente a distância de manhattan além de uma variação desta.

4.1.1 Distância de Manhattan/Geometria do Táxi

```
h((Cx,Cy),C):-
        estado_final((Fx,Fy)),
        (Cx>=Fx
                -> K1 is Cx-Fx,
                (Cy>=Fy
                        -> K2 is Cy-Fy,
                                C is K1 + K2
                        ;  K2 is Fy-Cy,
                                C is K1 + K2
                        )
                ; K1 is Fx-Cx,
                 (Cy>=Fy
                        -> K2 is Cy-Fy,
                                C is K1 + K2
                        ;  K2 is Fy-Cy,
                                C is K1 + K2
                        )
                ).
```

Esta heurística calcula a soma do módulo das distâncias entre as coordenadas X e Y de dois pontos distintos, resultando num valor que quanto menor for, mais optima é a distância entre dois pontos.

4.1.2 Heuristica Euclediana

```
h1((Cx,Cy),C):-
        estado_final((Fx,Fy)),
        (Cx>=Fx
                -> K1 is Cx-Fx,
                        K3 is K1*2,
                (Cy>=Fy
                        -> K2 is Cy-Fy,
                                C is K3 + K2
                        ;  K2 is Fy-Cy,
                                C is K3 + K2
                        )
                ; K1 is Fx-Cx,
                        K3 is K1*2,
                 (Cy>=Fy
                        -> K2 is Cy-Fy,
                                C is K3 + K2
                        ;  K2 is Fy-Cy,
                                C is K3 + K2
                        )
                ).
```

Esta heuristica, apesar de semelhante à primeira, dá prioridade ao movimento vertical, ou seja, duplica os valores das coordenadas horizontais, resultando assim um custo superior quando tem que se movimentar horizontalmente. Neste caso especifico, chegámos à conclusão que é uma heuristica mais eficiente que a distância de Manhattan original.

4.2 Alinea b)

Utilizámos o algoritmo de pesquisa informada A* para compararmos ambas as heurístiscas anteriores.

Heuristicas	Distância de Manhattan	Euclediana (h2)
Tempo PC Henrique	0.007s	0.003s
Tempo PC José	0.005s	0.003s
Custo	16	16

Através dos dados recolhidos na comparação de ambas as heurísticas em duas máquinas diferentes, podemos concluir que neste caso particular, apesar de ambas apresentarem o mesmo custo, a Heurística Euclediana é ligeiramente mais rápida a chegar à solução, logo podemos concluir que das duas, esta é mais eficiente do que a Distância de Manhattan.

4.3 alinea c)

Heuristicas	Distância de Manhattan	Euclediana (h2)
nós visitados	81	17
número máximo nós em memória	35	33

5 Conclusão

Com este trabalho conseguimos aprofundar um pouco o nosso conhecimento no comportamento e manipulação de vários algoritmos de pesquisa. Foi importante vermos e testarmos a forma como os algoritmos são mais ou menos eficientes pois apesar de sabermos que a optimização e eficiência se tratam de partes importantissimas da programação, na verdade nunca tínhamos explorado este tema de uma forma concreta.