UNIVERSIDADE DE ÉVORA

Inteligência Artificial
2015/2016

Resolução de problemas como problemas de pesquisa no espaço de estados

Docente:

Irene Pimenta Rodrigues

Discentes:

Pedro Jacob - 27421

André Figueira - 31626

Introdução

Neste primeiro trabalho da disciplina de Inteligencia Artificial pretendeu-se explorar os conceitos de pesquisa informada e não informada. Em paralelo pretendeu-se também explorar o porque de uso de heurísticas e a importância de a escolha de uma boa heurística que seja especializada para o problema em questão.

1. Definição do Problema

O problema é uma sala NxN em que um agente tem de ir desde o ponto A ao ponto B. Neste problema como descrito pelo enunciado:

- Matriz 30x30
- Origem: (18,18)
- Destino: (26,26)

1.1. Estado Inicial

O estado inicial corresponde a um só argumento que é a posição inicial, ou seja, origem do Agente.

```
% ----- estado inicial -

estado_inicial((18,18)).

% ---------------------
```

1.2. Estado final

O estado final corresponde a um só argumento que é a posição final, ou seja, o destino do Agente.

```
% ----- estado final --

estado_final((26,26)).

% ---------------------
```

1.3 Operações

As operações possíveis são mover para a esquerda, direita, subir e descer.

Andar para a esquerda implica movimentar-se menos uma unidade no eixo do X (X - 1).
Andar para a direita implica movimentar-se mais uma unidade no eixo do X (X + 1).
Subir implica movimentar-se mais uma unidade mais uma unidade no eixo do Y (Y + 1).
Descer implica movimentar-se menos uma unidade no eixo do Y (Y - 1).

```
% ---------------------------- OPERACOES ------------------------------------ %

op((X,Y),sobe,(X,Z),1):-
    Y<30,
    Z is Y+1,
    (not_been_here((X,Z)) -> \+verificar_bloqueio((X,Y),(X,Z))).

op((X,Y),dir,(Z,Y),1):-
    X<30,
    Z is X+1,
    (not_been_here((Z,Y)) -> \+verificar_bloqueio((X,Y),(Z,Y))).

op((X,Y),desce,(X,Z),1):-
    Y>1, Z is Y-1,
    (not_been_here((X,Z)) -> \+verificar_bloqueio((X,Y),(X,Z))).

op((X,Y),esq,(Z,Y),1):-
    X>1,
    Z is X-1,
    (not_been_here((Z,Y)) -> \+verificar_bloqueio((X,Y),(Z,Y))).

% --------------------------------------------------------------------------- %
```

Para um operação suceder, o movimento a realizar tem de estar dentro dos limites da Matriz. Se tal se verificar, verifica-se se a sala para onde se quer ir já foi visitada, se sim verifica não se existe bloqueio no caminho. Se todas se verificarem, então a operação é valida.

1.4 Nós visitados

Para evitar no programa “loops” decidiu-se então criar um predicado dinâmico que regista o nós que já foram visitados.

```
:-dynamic(has_been_visited/1).  % salas que ja foram visitadas
```

Sempre que é executada uma operação é verificado se a posição que quer ir já foi visitada. Se sim, a operação falha. Caso contrário, executa um asserta/1 que adiciona a sala à base de dados o nó.

1.5. Bloqueios

Bloqueios usados. Estes foram descritos no enunciado.

```
% --------- bloqueio --------- %

bloqueio((1,1),(1,2)).
bloqueio((1,2),(1,3)).
bloqueio((2,3),(2,2)).
bloqueio((3,4),(4,4)).
bloqueio((4,5),(3,5)).

% --------------------------- %
```

1.6. 1º Pergunta a)
Algoritmo de pesquisa não informada

O algoritmo que pensamos que resolve a situação em questão é o de pesquisa em profundidade. Isto porque em comparação aos outros algoritmos utilizados, verificou-se que:

O iterativo mostrou-se como o pior algoritmo dos 3. É o que contém menos nós em memória (cerca de 3x menos que largura e 2x menos que profundidade) mas em tempo de execução demora muito mais que os outros 2 isto porque passa por muito mais nós que qualquer um (cerca de 6x mais que o largura).

O de largura, mostrou-se como uma alternativa ao de profundidade, pois embora tenha visitado mais nós que profundidade (cerca de 4x mais) o número de nós em memória foi menor (cerca de 2x menos).

O do profundidade mostrou-se como o mais forte dos 3 no geral, isto porque visitou muito menos nós que o de largura (cerca de 4x menos) mas um ligeiro aumento de nós em memória (2x mais que largura). Mas o que nos fez decidir o profundidade foi mais na diminuição no número de nós visitados. Nota-se também, que a ordem das operações escolhida influenciou este algoritmo como esperado, obtendo assim melhores resultados para este problema em questão.

Portanto em suma, o de profundidade foi o escolhido pois era o que tinha melhores resultados dos 3 em geral.

1.7 1º Pergunta b) Número de nós visitados e em memória

Pesquisa iterativa:

Nós visitados: 2217
Máximo de nos em memoria: 30

Pesquisa em largura:

Nós visitados: 472
Máximo de nos em memoria: 50

Pesquisa em profundidade:

Nós visitados: 87
Máximo de nos em memoria: 93

2.

2.1. Duas heurísticas admissíveis para estimar o custo de um estado até a solução para este problema.

2.1.1. Heurística 1

Esta heurística calcula a distância entre o ponto actual e o destino, ou seja a distância euclídia. Sendo esta distância uma linha recta, indica que quanto menor for a distância, mais próximo estará do destino, significando que mais perto está da solução sendo esta heurística das melhores possíveis.

h1(PosActual, X)

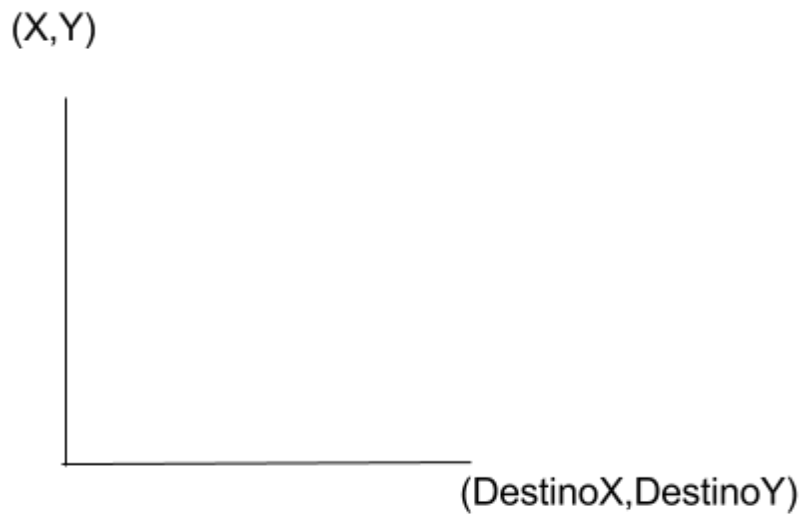
PosActual é a coordenada actual: (X,Y).
X é o valor da distância.

Representação da distância euclídia.

2.1.2. Heurística 2

Esta heurística é a distância de Manhattan. .

Nesta heurística ao contrário da primeira em que é distância em linha recta, a distância de Manhattan é a soma dos componentes horizontais e verticais, indicando o número de passos que faltam até ao destino. Quanto menor for esta distância, mais perto estará do destino.



Representação da distância de Manhattan.

2.1.2 Pergunta 2 b) Algoritmo e Heurística mais eficiente

O algoritmo mais eficiente que achamos que resolve o problema é o A*.
Em vários testes reparou-se que houve uma diminuição significativa nos nós visitados e nos nós em memória em relação aos de pesquisa não informada.

A heurística mais eficiente, para esta dimensão foi sem dúvida a segunda heurística, que é a distância de Manhattan, visto que em termos de nós visitados e nós em memória foi menor. Esta diferença foi pequena mas foi conclusiva o suficiente para escolher a segunda heurística.

2.1.3 Pergunta 2 c) Máximo de nós em memória e visitados

Usando a primeira heurística:

Nós visitados: 100
Máximo de nos em memoria: 37

Usando a segunda heurística:

Nós visitados: 81
Máximo de nos em memoria: 35

Conclusão

Ao fazer uma melhor observação dos resultados obtidos podemos facilmente concluir a importância extrema de escolher algoritmos para problemas individualmente, e que o uso de uma boa heurística é indispensável, visto que permite a redução de uso de recursos e tempo de execução.

É de referir também que a exposição ao uso de algoritmos de profundidade e largura permitiu entender que não há uma opção melhor que a outra, e que a escolha depende totalmente dos recursos disponíveis e a natureza do problema.