1º Trabalho de Inteligência Artificial

2018/2019

Resolução de problemas como problemas de pesquisa no espaço de estados

Docente:
Irene Pimenta Rodrigues

Realizado por:

Daniel Serrano – 35087

Miguel Serrano - 34149

15 de março de 2019

Conteúdo

Introdução:

Neste trabalho, no âmbito da cadeira de Inteligência Artificial iremos utilizador métodos de pesquisa informada e pesquisa não informada para resolver um exercício proposto pela professora. Exercício este que consiste na pesquisa de um caminho de um dado ponto até outro dado ponto, numa matriz NxN, e em que certos caminhos estão bloqueados.

1º Exercício:

Para representar o espaço de estados utilizámos um tuplo com as coordenadas:

```
estado_inicial((2,2)).
estado_final((8,8)).
```

Para representar as portas bloqueadas utilizámos pares de tuplos, com as coordenadas:

```
blocked((1,2), (1,3)).
blocked((1,3), (1,2)).
blocked((2,3), (2,2)).
blocked((2,2), (2,3)).
blocked((3,4), (4,4)).
blocked((4,4), (3,4)).
blocked((4,5), (3,5)).
blocked((3,5), (4,5)).
```

Para definir o tamanho da matriz utilizámos variáveis estáticas:

```
largura(8).
profundidade(8).
```

De modo a que não existissem loops, ou passagem desnecessárias por salas já visitadas, utilizámos uma definição Dynamic para registar as coordenadas:

```
:- dynamic(visited/1).
visited((2, 2)).
```

Para representar os operadores de transição, ou seja, mover para cima, baixo, direita ou esquerda, utilizámos os seguintes predicados, que caso a sala para onde se pretenda ir já tenho sido visitada, ou o caminho esteja bloqueado dá “fail” caso contrário o movimento é realizado e essa mesma sala será adicionada às salas já visitadas:

```
op((X, Y), baixo, (Z, Y), 1) :-
        profundidade(Prof),
        X < Prof,
        Z is X+1,
        ( visited((Z,Y))
            -> fail
            ; ( blocked((X, Y), (Z, Y))
                -> fail
                ; asserta(visited((Z,Y)))
              )
        ).


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

op((X, Y), esq, (X, Z), 1) :-
        Y > 1,
        Z is Y-1,
        ( visited((X,Z))
            -> fail
            ; ( blocked((X, Y), (X, Z))
                -> fail
                ; asserta(visited((X,Z)))
              )
         ).

op((X, Y), sobe, (Z, Y), 1) :-
        X > 1,
        Z is X-1,
        ( visited((Z,Y))
            -> fail
            ; ( blocked((X, Y), (Z, Y))
                -> fail
                ; asserta(visited((Z,Y)))
              )
        ).
```

1.a)

Para a resolução deste exercício temos de ter em conta vários pormenores, sendo um deles a ordem pela qual os movimentos são feitos. No caso do nosso trabalho optámos pela ordem ser baixo, direita, esquerda, cima, o que, tendo em conta que a posição inicial e a posição final vai fazer com que o algoritmo de pesquisa mais eficiente seja a pesquisa em profundidade, pois com este algoritmo vai descer até não ser possível descer mais (30) e depois começa a ir para a direita até chegar à posição final.

1.b)

Para realizar esta tarefa tivemos de modificar o código, pois o tamanho da matriz, assim como as posições iniciais e finais foram alteradas:

```
estado_inicial((18,18)).
estado_final((26,26)).
```

```
largura(30).
profundidade(30).
```

i. + ii.

Algoritmo de Profundidade:

```
Num. de nos visitados: 45
Num. maximo de nos em memoria: 54
```

Algoritmo de Largura:

```
Num. de nos visitados: 473
Num. maximo de nos em memoria: 50
```

Para testar estes procedimentos é necessário:

1- Abrir um terminal na pasta onde estão os ficheiros
2- Abrir o gprolog
3- Escrever [pni].
4- Escrever pesquisa('main.pl', profundidade) ou pesquisa('main.pl', largura)

2º Exercício:

2.a)

Primeira Heurística: Decidimos utilizar a distância de Manhattan em que se calcula a diferença entre as coordenadas finais e as iniciais, somam-se os valores e opta-se pelo caminho mais curto.

```
%Heuristica 1 - Distancia de manhattan
heu_1((Ix,Iy),SOMA):-
    estado_final((Fx,Fy)),
    Dx is abs(Ix - Fx),
    Dy is abs(Iy - Fy),
    SOMA is Dx + Dy.
```

Segunda Heurística: Decidimos utilizar a distância euclidiana que, acaba por ser parecida à primeira, mas, neste caso calcula-se a raiz quadrada da soma dos quadrados da diferença entre as coordenadas finais e as iniciais.

```
%Heuristica 2 - Distancia euclidiana
heu_2((Ix,Iy),SOMA):-
    estado_final((Fx,Fy)),
    Dx is abs(Ix - Fx),
    Dy is abs(Iy - Fy),
    SOMA is round(sqrt(Dy ** 2 + Dx ** 2)).
```

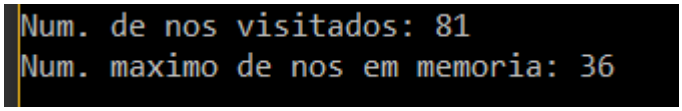
2.b)

O código em Prolog, do algoritmo de pesquisa informada que utilizámos para resolver este problema foi o código utilizado nas aulas práticas, código esse que consiste no algoritmo de pesquisa A* e que, como estudámos, é um algoritmo bom para este tipo de problemas.

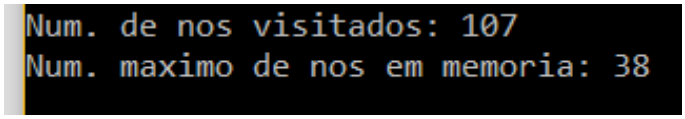
2.c)

i. + ii.

Primeira heurística:



Segunda heurística:



Com estes resultados podemos então concluir que a primeira heurística foi a melhor heurística utilizado este algoritmo de pesquisa (pesquisa A*).

Para testar estes procedimentos é necessário:

1- Abrir um terminal na pasta onde estão os ficheiros
2- Abrir o gprolog
3- Escrever [pi].
4- Escrever pesquisa('main.pl', a).

Conclusão:

Com a realização deste trabalho ficámos a ter mais conhecimento sobre os tipos de pesquisa informada e não informada o que, no futuro, penso que estes conhecimentos nos irão dar bastante jeito.
Na segunda parte do trabalho em que tivemos de implementar duas heurísticas tivemos um bocado dificuldade pois não percebíamos muito bem este conceito de heurística mas com o estudo e com a realização do trabalho conseguis adquirir esse conhecimento.