UNIVERSIDADE DE ÉVORA

1º Trabalho de Inteligência Artificial

Yaroslav Kolodiy `139859` Eduardo Medeiros `139873`

Janeiro, Ano Letivo 2019/2020

Inteligência Artificial

Prof. Paulo Quaresma

Índice

1 Respostas

1.1 Nota Prévia

Nota prévia: Ambos os programas podem ser executados com o predicado

$$pesquisa(L, X, Y, EI, EF).$$

cujo os termos representam respetivamente:

1. A lista de portas fechadas/restrições.
2. O tamanho do tabuleiro me X.
3. O tamanho do tabuleiro em Y.
4. Lista que representa o estado inicial [XI,YI].
5. Lista que represneta o estado final [XF,YF].

Para o exemplo do enunciado o predicado seria o seguinte:

$$pesquisa([[1,1,1,2],[2,1,2,2],[3,1,4,1],[3,2,3,3],[4,2,4,3]],4,4,[1,1],[4,4]).$$

1.2 Respostas

1. Representado em 2.1.

   (a) O algoritmo usado foi o de pesquisa em profundidade sem ciclos, pois este evita que nós previamente visitados sejam visitados novamente, evitando assim expansões desnecessárias, pois estas já foram feitas previamente.

   Uma causa que pode afetar o desempenho do algoritmo usado é a ordem/precedência das operações (cima, baixo, esquerda, direita). A ordem escolhida pelo grupo foi respetivamente, Cima, Esquerda, Baixo, Direita, de modo a que o número de nós visitados seja menor neste respetivo caso.

   (b) i. No total foram visitados 10 estados.

   ii. O número máximo de estados simultaneamente em memória foi 9, tendo em conta o algoritmo implementado.

2. (a) Uma heurística admissível para estimar o custo de um estado até à solução do problema, é o cálculo da distancia do estado atual até ao estado da solução. De modo a calcular essa distância, foi considerado pelo grupo, que a melhor abordagem é o uso Teorema de Pitágoras, i.e.

$$distancia = \sqrt{C1^2 + C2^2}$$

uma vez que sabendo as coordenadas X e Y dos dois estados, pode ser facilmente calculada a distância.

(b) Representado em 2.2.

(c) i. No total foram visitados 16 estados.

ii. O número máximo de estados simultaneamente em memória foram 20.

2 Anexos

2.1 Anexo 1 - Resposta pergunta 1

```
%Descricao do problema:
/*
    Dando um estado inicial [XI, YI], tenta ir para um
       estado final [XF, YF]
    tendo em conta as restri   es impostas, i.e. as
       tranci   es entre estados
    que n o pode efetuar.
*/

:- dynamic(fechado/4).
:- dynamic(tamanhoX/1).
:- dynamic(tamanhoY/1).
:- dynamic(estado_inicial/1).
:- dynamic(estado_final/1).

criar_fecho([]).
criar_fecho([[X,Y,X1,Y1]|T]):-
    asserta(fechado(X,Y,X1,Y1)),
    asserta(fechado(X1,Y1,X,Y)),
    criar_fecho(T).

pesquisa(R,Tx,Ty,Ei,Ef):-
    criar_fecho(R),
    asserta(tamanhoX(Tx)),
    asserta(tamanhoY(Ty)),
    asserta(estado_inicial(Ei)),
    asserta(estado_final(Ef)),
    pesquisa.


%estado_inicial(Estado)
estado_inicial([1,1]).

%estado_final(Estado)
estado_final([4,4]).

%restricoes
```

```
%fechado(X_Atual, Y_Atual, X_Destino, Y_Destino)
fechado(1,1,1,2).
fechado(1,2,1,1).

fechado(2,1,2,2).
fechado(2,2,2,1).

fechado(3,1,4,1).
fechado(4,1,3,1).

fechado(3,2,3,3).
fechado(3,3,3,2).

fechado(4,2,4,3).
fechado(4,3,4,2).

%tamanho_tabela
tamanhoX(4).
tamanhoY(4).

%representacao dos operadores
%op(Eact,OP,Eseg,Custo)

/*
    Para todos os predicados op/4.

    Verifica se est  nos limites do tabuleiro,
    e se a transi  o entre Eact e o Eseg    poss vel
       .
*/

op([X,Y], cima, [X1,Y], 1) :-
    X > 1,
    X1 is X - 1,
    \+ fechado(X,Y,X1,Y).

op([X,Y], esquerda, [X,Y1], 1) :-
    Y > 1,
    Y1 is Y - 1,
    \+ fechado(X,Y,X,Y1).
```

```
op([X,Y], baixo, [X1,Y], 1) :-
    tamanhoX(T),
    X < T,
    X1 is X + 1,
    \+ fechado(X,Y,X1,Y).

op([X,Y], direita, [X,Y1], 1) :-
    tamanhoY(T),
    Y < T,
    Y1 is Y + 1,
    \+ fechado(X,Y,X,Y1).

%representacao dos nos
%no(Estado, no_pai, Operador, Custo, Profundidade)

pesquisa_aux([no(E,Pai,Op,C,P)|_],no(E,Pai,Op,C,P), _)
  :-
        estado_final(E).
pesquisa_aux([no(E,Pai,Op,C,P)|R],Sol, LE):-
    %write(E),
        \+ member(E, LE),
        expande(no(E,Pai,Op,C,P),Lseg),
        insere_inicio(Lseg,R,LFinal),
        %length(LFinal, Tamanho),
        %write(Tamanho), nl,
        pesquisa_aux(LFinal,Sol, [E|LE]).
pesquisa_aux([no(E,_,_,_,_)|R],Sol, LE):-
        member(E, LE),
    pesquisa_aux(R,Sol, LE).


expande(no(E,Pai,Op,C,P),L):-
        findall(no(En,no(E,Pai,Op,C,P), Opn, Cnn, P1),
                (op(E,Opn,En,Cn), P1 is P+1, Cnn is Cn+
                   C),
                L).

pesquisa :-
        estado_inicial(S0),
        pesquisa_aux([no(S0,[],[],0,0)], S, []),
        write(S), nl.
```

```
insere_inicio(A,B,C) :- append(A, B, C).

insere_fim(A,B,C) :- append(B, A, C).
```

2.2 Anexo 2 - Resposta pergunta 2.b

```
%Descricao do problema:
/*
    Dando um estado inicial [XI,YI], tenta ir para um
       estado final [XF, YF]
    tendo em conta as restri   es impostas, i.e. as
       tranci   es entre estados
    que n o pode efetuar.
*/

:- dynamic(fechado/4).
:- dynamic(tamanhoX/1).
:- dynamic(tamanhoY/1).
:- dynamic(estado_inicial/1).
:- dynamic(estado_final/1).

criar_fecho([]).
criar_fecho([[X,Y,X1,Y1]|T]):-
    asserta(fechado(X,Y,X1,Y1)),
    asserta(fechado(X1,Y1,X,Y)),
    criar_fecho(T).

pesquisa(R,Tx,Ty,Ei,Ef):-
    criar_fecho(R),
    asserta(tamanhoX(Tx)),
    asserta(tamanhoY(Ty)),
    asserta(estado_inicial(Ei)),
    asserta(estado_final(Ef)),
    pesquisa.

%Estados representados por listas [X,Y].

/*%estado_inicial(Estado)
estado_inicial([1,1]).

%estado_final(Estado)
estado_final([4,4]).

%restricoes
%fechado(X_Atual,Y_Atual,X_Destino,Y_Destino)
```

```
fechado(1,1,1,2).
fechado(1,2,1,1).

fechado(2,1,2,2).
fechado(2,2,2,1).

fechado(3,1,4,1).
fechado(4,1,3,1).

fechado(3,2,3,3).
fechado(3,3,3,2).

fechado(4,2,4,3).
fechado(4,3,4,2).

%tamanho_tabela
tamanhoX(4).
tamanhoY(4).*/

%representacao dos operadores
%op(Eact,OP,Eseg,Custo)

/*
    Para todos os predicados op/4.

    Verifica se est  nos limites do tabuleiro,
    e se a transi  o entre Eact e o Eseg    poss vel
       .
*/

op([X,Y], cima, [X1,Y], 1) :-
    X > 1,
    X1 is X - 1,
    \+ fechado(X,Y,X1,Y).

op([X,Y], baixo, [X1,Y], 1) :-
    tamanhoX(T),
    X < T,
    X1 is X + 1,
    \+ fechado(X,Y,X1,Y).
```

```
op([X,Y], esquerda, [X,Y1], 1) :-
    Y > 1,
    Y1 is Y - 1,
    \+ fechado(X,Y,X,Y1).

op([X,Y], direita, [X,Y1], 1) :-
    tamanhoY(T),
    Y < T,
    Y1 is Y + 1,
    \+ fechado(X,Y,X,Y1).

%representacao dos nos
%no(Estado, no_pai, Operador, Custo, Heuristica,
   Profundidade)
absoluto(X,X) :- X >= 0, !.
absoluto(X,Y) :- X < 0, Y is -X.

heuristica([XInicial, YInicial] , H):- estado_final([
   XFinal, YFinal]),
    X is XFinal-XInicial, absoluto(X, XCalculado),
    Y is YFinal-YInicial, absoluto(Y, YCalculado),
    H is XCalculado+YCalculado.


pesquisa_aux([no(E,Pai,Op,C,H,P)|_],no(E,Pai,Op,C,H,P))
    :-
        estado_final(E).
pesquisa_aux([E|R],Sol):-
    %length(R, Tamanhao),
    %write(Tamanhao), nl,
        expande(E,Lseg),
        insere_ordenado(Lseg,R,LFinal),
        pesquisa_aux(LFinal,Sol).

expande(no(E,Pai,Op,C,H,P),L):-
        findall(no(En,no(E,Pai,Op,C,H,P), Opn, Cnn, H1,
            P1),
                (op(E,Opn,En,Cn),
                    P1 is P+1,
                    Cnn is Cn+C,
                    heuristica(En,HCalculada),
```

```
                    H1 is Cnn + HCalculada),
                L).

pesquisa :-
        estado_inicial(S0),
        pesquisa_aux([no(S0,[],[],0,0,0)], S),
        write(S), nl.


ins_ord(E, [], [E]).
ins_ord(no(E,Pai,Op,C,Heur,P), [no(E1,Pai1,Op1,C1,Heur1
   ,P1)|T], [no(E,Pai,Op,C,Heur,P),no(E1,Pai1,Op1,C1,
   Heur1,P1)|T]) :-
    Heur =< Heur1.
ins_ord(no(E,Pai,Op,C,Heur,P), [no(E1,Pai1,Op1,C1,Heur1
   ,P1)|T], [no(E1,Pai1,Op1,C1,Heur1,P1)|T1]) :-
        ins_ord(no(E,Pai,Op,C,Heur,P), T, T1).

insere_ordenado([],L,L).
insere_ordenado([A|T], L, LF):-
        ins_ord(A,L,L1),
        insere_ordenado(T, L1, LF).
```