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: Os programa podem ser executado com o predicado

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

caso queira usar termos específicos. Estes representam respectivamente:

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].

Tambem pode ser usado o predicado

$$pesquisa.$$

no qual serão usados os predicados "embutidos"no código.

Para o exemplo do enunciado o predicado seria o seguinte:

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

1.2 Respostas

1. Representado em 2.1.

   (a) O algoritmo usado foi o de pesquisa local com deteção de ciclos, pois este evita que nós previamente visitados sejam visitados novamente, evitando assim novas expansões desnecessárias.

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

   Uma outra causa que afeta o resultado final é heurística escolhida. Se o custo para atingir qualquer estado for de 1, o número de estados visitados, no exemplo do enunciado e com a ordem escolhida, é

menor. No entanto com a heurística escolhida pelo grupo, em que o estado final tem custo 0, os estado do centro do tabuleiros têm custo 1, os dos limites têm custo 2 e, mesmo não havendo ciclos, o estado inicial tem custo 3, deste modo, no exemplo enunciado, o número de estados visitados é ligeiramente maior do que no caso anterior.

(b) i. A. Com o custo de todos os estados a 1.
[1,1] baixo
[2,1] baixo
[3,1] direita
[3,2] cima
[2,2] direita
[2,3] baixo
[3,3] direita
[3,4] cima
[2,4] cima
[1,4]

B. Com o custo do estado final a 0, estados do meio do tabuleiro a 1, estados dos limites a 2 e estado inicial a 3.
[1,1] baixo
[2,1] baixo
[3,1] direita
[3,2] cima
[2,2] cima
[1,2] direita
[1,3] direita
[1,4]

ii. Tendo em conta a resposta à pergunta anterior, os caminhos têm tamanho 8 e 10, respectivamente.

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(tamanho/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,T,Ei,Ef):-
    criar_fecho(R),
    asserta(tamanho(T)),
    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([1,4]).

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

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) :-
    tamanho(T),
    X < T,
    X1 is X + 1,
    \+ fechado(X,Y,X1,Y).

%representacao dos nos
%no(Estado,no_pai,Operador,Custo,Heuristica,
   Profundidade)
heur([X,Y], 0) :- estado_final([X,Y]).
heur([X,Y], 3) :- estado_inicial([X,Y]).
heur([X,Y], 1) :- tamanho(T), X > 1, X < T, Y > 1, Y <
   T.
%heur([_,_], 1).
heur([T,_], 2) :- tamanho(T).
heur([_,T], 2) :- tamanho(T).
heur([1,_], 2).
heur([_,1], 2).

pesquisa_local_hill_climbingSemCiclos(E, _) :-
    estado_final(E),
    write(E), write('␣').

pesquisa_local_hill_climbingSemCiclos(E, L) :-
    write(E), write('␣'),
    expande(E,LSeg),
    sort(3, @=<, LSeg, LOrd),
    obtem_no(LOrd, no(ES, Op, _)),
    \+ member(ES, L),
    write(Op), nl,
    (pesquisa_local_hill_climbingSemCiclos(ES,[E|L]) ;
       write(undo(Op)), write('␣'), fail).

expande(E, L):-
    findall(no(En,Opn, Heur),
                (op(E,Opn,En,_), heur(En, Heur)),
                L).
```

```
obtem_no([H|_], H).
obtem_no([_|T], H1) :-
    obtem_no(T, H1).

pesquisa :-
    estado_inicial(S0),
    pesquisa_local_hill_climbingSemCiclos(S0, []).
```