2º Trabalho de Inteligência Artificial

2018/2019

UNIVERSIDADE DE ÉVORA

Resolução do jogo do quadrado mágico e do sudoku como problemas CSP

Docente:
Irene Pimenta Rodrigues

Realizado por:

Daniel Serrano – 35087

Miguel Serrano - 34149

24 de março de 2019

Conteúdo

Introdução: 3
1º Exercício: 4
1.a) 4
1.b) 5
1.c) 6
2º Exercício: 7
2.a) 7
2.b) 9
2.c) 9
Conclusão: 10

Introdução:

Neste trabalho, no âmbito da cadeira de Inteligência Artificial iremos resolver o jogo do quadrado mágico e do sudoku abordando estes jogos como problemas de satisfação de restrições, e, utilizando as pesquisas de backtrack e forwardcheking.

1º Exercício:

1.a)

Para representar este problema como um problema de satisfação de restrições, considerámos que cada variável tem uma posição, um domínio e o valor da respetiva posição. De referir que o domínio no nosso exemplo vai de 1 a 9 pois o quadrado é de dimensão 3x3 (`size(3).`):

```
estado_inicial( e([v(n(1,1),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(1,2),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(1,3),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(2,1),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(2,2),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(2,3),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(3,1),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(3,2),[1,2,3,4,5,6,7,8,9],_Val),
            v(n(3,3),[1,2,3,4,5,6,7,8,9],_Val)],[]) ).
```

Em relação às restrições o nosso código começa por verificar se todos os elementos do tabuleiro são diferentes:

```
restricoes(e(NAfect,Afect)):- all_diff(Afect), valida_somas(Afect).
```

```
all_diff([]).
all_diff([v(_,_,V)|Afect]):- member(v(_,_,V),Afect),!,fail.
all_diff([_|Afect]):- all_diff(Afect).
```

De seguida verifica as somas de todas as colunas, linhas e diagonais do tabuleiro, comparando essas somas com o número 15 pois é a solução correta de um tabuleiro com a dimensão 3x3:

```
valida_somas(L):- % linhas
                findall(V,member(v(n(1,_),_,V), L),L1), sum_total(L1),
                findall(V,member(v(n(2,_),_,V), L),L2), sum_total(L2),
                findall(V,member(v(n(3,_),_,V), L),L3), sum_total(L3),

                %colunas
                findall(V,member(v(n(_,1),_,V), L),C1), sum_total(C1),
                findall(V,member(v(n(_,2),_,V), L),C2), sum_total(C2),
                findall(V,member(v(n(_,3),_,V), L),C3), sum_total(C3),

                % diagonal X = y
                findall(V,member(v(n(1,1),_,V),L),D1),
                findall(V,member(v(n(2,2),_,V), L),D2),
                findall(V,member(v(n(3,3),_,V), L),D3),append(D1,D2,M),append(M,D3,X),sum_total(X),

                % diagonal
                findall(V,member(v(n(0,2),_,V),L),D4),
                findall(V,member(v(n(1,1),_,V), L),D5),
                findall(V,member(v(n(2,0),_,V), L),D6),append(D4,D5,P),append(P,D6,J),sum_total(J).

sum_total( [V1,V2,V3] ):-!, 15 is  V1+V2+V3.

sum_total(_).
```

O operador sucessor utilizado foi o mesmo que foi utilizado durante as aulas práticas:

```
sucessor(e([v(N,D,_)|R],E),e(R,[v(N,D,V)|E])):- member(V,D).
```

1.b)

Neste exercício era proposto resolver o problema com o algoritmo backtracking, um algoritmo de pesquisa não informada para CSPs e que se revelou bastante eficiente na resolução deste exercício:

```
b:- consult(quadrado_magico),
    estado_inicial(E0),
    back(E0,A),
    esc(A).

back(e([],A),A).
back(E,Sol):- sucessor(E,E1),
          restricoes(E1),
              back(E1,Sol).
```

Os resultados obtidos para este algoritmo foram os seguintes:

```
2 . 4 . 9
6 . 8 . 1
7 . 3 . 5
```

```
true ? ;
5
2 . 6 . 7
4 . 8 . 3
9 . 1 . 5
```

```
true ? ;
5
2 . 7 . 6
9 . 5 . 1
4 . 3 . 8
```

```
true ? ;
5
2 . 7 . 6
9 . 5 . 1
4 . 3 . 8
```

1.c)

Neste exercício era proposto resolver o problema com o algoritmo forward checking, um algoritmo que utiliza o algoritmo utilizado no exercício 1.b) mas que o completa verificando certos valores que ainda não foram definidos:

```
f:-consult(quadrado_magico),
        estado_inicial(E0),
        back(E0,A),
        write(A),nl,nl,esc(A).

back(e([],A),A).
back(E,Sol):- sucessor(E,E1),
            restricoes(E1),
            forwardC(E1,E2),
                back(E2,Sol).


sucessor(e([v(N,D,_)|R],E),e(R,[v(N,D,V)|E])):- member(V,D).

%ForwardChecking

forwardC(e(NAfect,[v(N,D,V)|Afect]),e(NAfectS,[v(N,D,V)|Afect])):-
        actualizaDom(V, NAfect, NAfectS).

actualizaDom(_,[],[]).
actualizaDom(V,[v(N,D,_)|NAfect],[v(N,DS,_)|NAfectS]):-
    delete(D,V,DS),
    actualizaDom(V, NAfect, NAfectS).
```

Os resultados obtidos para este algoritmo foram semelhantes ao anterior pois na execução deste jogo a diferença de eficiência não é notória:

```
2 . 4 . 9
6 . 8 . 1
7 . 3 . 5
```

```
true ? ;
5
[v(n(3,3),[5],5),v
,3,4,5,7,8,9],7),v

2 . 6 . 7
4 . 8 . 3
9 . 1 . 5
```

```
true ? ;
5
[v(n(3,3),[8]
,3,4,5,6,8,9]

2 . 7 . 6
9 . 5 . 1
4 . 3 . 8
```

```
true ? ;
8
[v(n(3,3),[8
,3,4,5,6,7,8

2 . 9 . 4
7 . 5 . 3
6 . 1 . 8
```

2º Exercício:

2.a)

Igualmente ao feito no primeiro exercício (quadrado mágico), iremos representar este problema como um problema de satisfação de restrições com o uso de varáveis que assumem uma posição, um domínio, e o valor da respetiva posição. De referir que neste caso, os valores das posições de cada variável (X,Y) varia entre 1-9 devido à dimensão do tabuleiro ser 9x9 (`size(9).`), e, o domínio das variáveis varia também entre 1-9 pois tratamos quadrantes de dimensão 3x3.

Na imagem seguinte irei apresentar como foram representados os estados iniciais:

```
estado_inicial(e([v(n(1,1),[1,2,3,4,5,6,7,8,9],_),v(n(1,3),[1,2,3,4,5,6,7,8,9],_),v(n(1,4),[1,2,3,4,5,6,7,8,9],_),v(n(1,5),[1,2,3,4,5,6,7,8,9],_),v(n(1,7),[1,2,3,4,5,6,7,8,9],_),
                 v(n(2,1),[1,2,3,4,5,6,7,8,9],_),v(n(2,2),[1,2,3,4,5,6,7,8,9],_),v(n(2,3),[1,2,3,4,5,6,7,8,9],_),v(n(2,5),[1,2,3,4,5,6,7,8,9],_),v(n(2,7),[1,2,3,4,5,6,7,8,9],_),
                 v(n(2,8),[1,2,3,4,5,6,7,8,9],_),v(n(2,9),[1,2,3,4,5,6,7,8,9],_),v(n(3,2),[1,2,3,4,5,6,7,8,9],_),v(n(3,3),[1,2,3,4,5,6,7,8,9],_),v(n(3,4),[1,2,3,4,5,6,7,8,9],_),
                 v(n(3,5),[1,2,3,4,5,6,7,8,9],_),v(n(3,6),[1,2,3,4,5,6,7,8,9],_),v(n(3,8),[1,2,3,4,5,6,7,8,9],_),v(n(4,1),[1,2,3,4,5,6,7,8,9],_),v(n(4,2),[1,2,3,4,5,6,7,8,9],_),
                 v(n(4,3),[1,2,3,4,5,6,7,8,9],_),v(n(4,4),[1,2,3,4,5,6,7,8,9],_),v(n(4,5),[1,2,3,4,5,6,7,8,9],_),v(n(4,7),[1,2,3,4,5,6,7,8,9],_),v(n(4,8),[1,2,3,4,5,6,7,8,9],_),
                 v(n(4,9),[1,2,3,4,5,6,7,8,9],_),v(n(5,1),[1,2,3,4,5,6,7,8,9],_),v(n(5,2),[1,2,3,4,5,6,7,8,9],_),v(n(5,3),[1,2,3,4,5,6,7,8,9],_),v(n(5,4),[1,2,3,4,5,6,7,8,9],_),
                 v(n(5,7),[1,2,3,4,5,6,7,8,9],_),v(n(6,2),[1,2,3,4,5,6,7,8,9],_),v(n(6,3),[1,2,3,4,5,6,7,8,9],_),v(n(6,5),[1,2,3,4,5,6,7,8,9],_),v(n(6,6),[1,2,3,4,5,6,7,8,9],_),
                 v(n(6,7),[1,2,3,4,5,6,7,8,9],_),v(n(6,8),[1,2,3,4,5,6,7,8,9],_),v(n(6,9),[1,2,3,4,5,6,7,8,9],_),v(n(7,1),[1,2,3,4,5,6,7,8,9],_),v(n(7,2),[1,2,3,4,5,6,7,8,9],_),
                 v(n(7,3),[1,2,3,4,5,6,7,8,9],_),v(n(7,5),[1,2,3,4,5,6,7,8,9],_),v(n(7,6),[1,2,3,4,5,6,7,8,9],_),v(n(7,7),[1,2,3,4,5,6,7,8,9],_),v(n(7,8),[1,2,3,4,5,6,7,8,9],_),
                 v(n(7,9),[1,2,3,4,5,6,7,8,9],_),v(n(8,3),[1,2,3,4,5,6,7,8,9],_),v(n(8,4),[1,2,3,4,5,6,7,8,9],_),v(n(8,6),[1,2,3,4,5,6,7,8,9],_),v(n(8,7),[1,2,3,4,5,6,7,8,9],_),
                 v(n(8,9),[1,2,3,4,5,6,7,8,9],_),v(n(9,1),[1,2,3,4,5,6,7,8,9],_),v(n(9,2),[1,2,3,4,5,6,7,8,9],_),v(n(9,4),[1,2,3,4,5,6,7,8,9],_),v(n(9,5),[1,2,3,4,5,6,7,8,9],_),
                 v(n(9,7),[1,2,3,4,5,6,7,8,9],_),v(n(9,8),[1,2,3,4,5,6,7,8,9],_),v(n(9,9),[1,2,3,4,5,6,7,8,9],_)],
```

De seguida, apresento as variáveis já preenchidas:

```
[v(n(1,2),[1,2,3,4,5,6,7,8,9],1),v(n(1,6),[1,2,3,4,5,6,7,8,9],8),v(n(1,8),[1,2,3,4,5,6,7,8,9],7),v(n(1,9),[1,2,3,4,5,6,7,8,9],3),v(n(2,4),[1,2,3,4,5,6,7,8,9],5),
v(n(2,6),[1,2,3,4,5,6,7,8,9],9),v(n(3,1),[1,2,3,4,5,6,7,8,9],7),v(n(3,7),[1,2,3,4,5,6,7,8,9],9),v(n(3,9),[1,2,3,4,5,6,7,8,9],4),v(n(4,6),[1,2,3,4,5,6,7,8,9],4),
v(n(5,5),[1,2,3,4,5,6,7,8,9],3),v(n(5,6),[1,2,3,4,5,6,7,8,9],5),v(n(5,8),[1,2,3,4,5,6,7,8,9],1),v(n(5,9),[1,2,3,4,5,6,7,8,9],8),v(n(6,1),[1,2,3,4,5,6,7,8,9],8),
v(n(6,4),[1,2,3,4,5,6,7,8,9],9),v(n(7,4),[1,2,3,4,5,6,7,8,9],7),v(n(8,1),[1,2,3,4,5,6,7,8,9],2),v(n(8,2),[1,2,3,4,5,6,7,8,9],6),v(n(8,5),[1,2,3,4,5,6,7,8,9],4),
v(n(8,8),[1,2,3,4,5,6,7,8,9],3),v(n(9,3),[1,2,3,4,5,6,7,8,9],5),v(n(9,6),[1,2,3,4,5,6,7,8,9],3)])).
```

Para representar as restrições, em primeiro foi utilizado um método semelhante ao do exercício anterior em que verificamos as linhas e as colunas, mas, sem a parte em que calculávamos as somas dos valores. Adicionámos ainda, uma restrição denominada “ver_quadrantes” que verifica se todos os 9 quadrantes, ou seja, “mini” quadrados 3x3 do tabuleiro têm todos os valores diferentes:

```
restricoes(e(NAfect,Afect)):-
    ver_linhas(e(Nafect,Afect)),
    ver_colunas(e(Nafect,Afect)),
    ver_quadrantes(e(Nafect,Afect)).


all_diff([]).
all_diff([H|T]) :-
    \+member(H, T), all_diff(T).

ver_linhas(e(Nafect,[v(n(X,Y), D, V)|R])):-
        findall(V1,member(v(n(X,_),_,V1),R),L),
        all_diff([V|L]).

ver_colunas(e(Nafect,[v(n(X,Y), D, V)|R])):-
        findall(V1,member(v(n(_,Y),_,V1),R),L),
        all_diff([V|L]).
```

```
ver_quadrantes(e(_, Afect)):-
    ver_quadrante(Afect, 1, 1, 3, Q1), all_diff(Q1),
    ver_quadrante(Afect, 1, 4, 6, Q2),all_diff(Q2),
    ver_quadrante(Afect, 1, 7, 9, Q3),all_diff(Q3),
    ver_quadrante(Afect, 4, 1, 3, Q4),all_diff(Q4),
    ver_quadrante(Afect, 4, 4, 6, Q5),all_diff(Q5),
    ver_quadrante(Afect, 4, 7, 9, Q6),all_diff(Q6),
    ver_quadrante(Afect, 7, 1, 3, Q7),all_diff(Q7),
    ver_quadrante(Afect, 7, 4, 6, Q8),all_diff(Q8),
    ver_quadrante(Afect, 7, 7, 9, Q9),all_diff(Q9).

ver_quadrante(L, X, Y, Y2, L2):-
    Y = Y2, X1 is X+2,
    ver_q_c(L, X, Y, X1, L2).

ver_quadrante(L, X, Y, Y2, L3):-
    Y < Y2, Y1 is Y+1,
    X1 is X+2,
    ver_q_c(L, X, Y, X1, L1),
    append(L1, L2, L3),
    ver_quadrante(L, X, Y1, Y2, L2).

ver_q_c(L, X, Y, X2, []):-
    X = X2,
    \+member(v(n(X, Y), _, _), L).

ver_q_c(L, X, Y, X2, [V]):-
        X = X2,
        member(v(n(X,Y), _, V), L).

ver_q_c(L, X, Y, X2, T):-
        X < X2, X1 is X+1,
        \+member(v(n(X, Y), _, _), L),
    ver_q_c(L, X1, Y, X2, T).

ver_q_c(L, X, Y, X2, [V|T]):-
    X < X2,
        member(v(n(X,Y), _, V), L),
    X1 is X+1,
        ver_q_c(L, X1, Y, X2, T).
```

2.b)

Neste exercício era proposto resolver o problema com o algoritmo backtracking, um algoritmo de pesquisa não informada para CSPs e que se revelou bastante eficiente na resolução do exercício anterior, mas, neste caso, apesar de não o conseguir comprovar, demorou significativamente mais tempo, no entanto irei apresentar de seguida a resolução obtida:

```
5 . 1 . 9 . 4 . 2 . 8 . 6 . 7 . 3
6 . 3 . 4 . 5 . 7 . 9 . 1 . 8 . 2
7 . 2 . 8 . 3 . 1 . 6 . 9 . 5 . 4
3 . 5 . 2 . 1 . 8 . 4 . 7 . 9 . 6
9 . 7 . 6 . 2 . 3 . 5 . 4 . 1 . 8
8 . 4 . 1 . 9 . 6 . 7 . 3 . 2 . 5
4 . 9 . 3 . 7 . 5 . 2 . 8 . 6 . 1
2 . 6 . 7 . 8 . 4 . 1 . 5 . 3 . 9
1 . 8 . 5 . 6 . 9 . 3 . 2 . 4 . 7
```

```
(204547 ms) no
```

Como se pode observar com estes resultados, apenas foi possível obter uma resolução para este exercício.

2.c)

Neste exercício era proposto resolver o problema com o algoritmo forward checking, um algoritmo que utiliza o algoritmo utilizado no exercício 1.b) mas que o completa verificando certos valores que ainda não foram definidos, no entanto, ao utilizar o algoritmo funcional do exercício anterior, não conseguis obter nenhum resultado:

```
(31 ms) no
```

Conclusão:

Com a realização deste trabalho ficámos a ter mais conhecimento sobre os tipos de backtracking e forward checking o que, no futuro, penso que estes conhecimentos nos irão dar bastante jeito.

No entanto, tivemos bastantes dificuldades a implementar o algoritmo de forward checking, principalmente de modo a que funcionasse com o problema do sudoku.