2º Trabalho Prático
Constraint Satisfaction Problems

Licenciatura em Eng. Informática

Inteligência Artificial

UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

André Baião 48092
Gonçalo Barradas 48402
Guilherme Grilo 48921

Docente: Irene Pimenta

1 Considere o problema do quadrado magico (3X3, 4X4, ... como um problema de CSP. Num quadrado magico todos os elementos são diferentes e a soma das linhas, das colunas e das duas diagonais principais são iguais

a) Represente este problema como um problema de satisfação de restrições em prolog.

```
1         % Estados: e(Lista de variaveis não instanciadas, Lista de posi
              ções instanciadas).
2         % Posição: p(Numero da posição no tabuleiro)
3         %Variaveis: v(p(Posição), Dominio, Valor)
4
5         %Estado Inicial
6         estado_inicial(e([
7             v(p(1),[1,2,3,4,5,6,7,8,9],_),
8             v(p(2),[1,2,3,4,5,6,7,8,9],_),
9             v(p(3),[1,2,3,4,5,6,7,8,9],_),
10            v(p(4),[1,2,3,4,5,6,7,8,9],_),
11            v(p(5),[1,2,3,4,5,6,7,8,9],_),
12            v(p(6),[1,2,3,4,5,6,7,8,9],_),
13            v(p(7),[1,2,3,4,5,6,7,8,9],_),
14            v(p(8),[1,2,3,4,5,6,7,8,9],_),
15            v(p(9),[1,2,3,4,5,6,7,8,9],_)],[])).
16
17        % Verificar restrições
18        igual(X,X).
19        % Soma total de uma linha, coluna ou diagonal.
20        doSoma(_,[],0).
21        doSoma(LE, [H|T], S):- member(v(p(H),_,V), LE),
22        doSoma(LE,T, S1), S is S1 + V.
23        % verifica as restinçoes com o auxilio do doSoma e do
              ve_repetidos
24        ve_restricoes(e([],L)):- ve_repetidos(L),
25        findall(LL, linhas(LL), [L1,L2,L3]),
26        findall(C, colunas(C), [C1,C2,C3]),
27        findall(D, diagonais(D), [D1,D2]),
28        doSoma(L,L1,SL1), doSoma(L,L2,SL2), doSoma(L,L3,SL3),
29        doSoma(L,C1,SC1), doSoma(L,C2,SC2), doSoma(L,C3,SC3),
30        doSoma(L,D1,SD1), doSoma(L,D2,SD2), igual(SL1,SL2),
31        igual(SL1,SL3),igual(SL1,SC1),igual(SL1,SC2),igual(SL1,SC3),
32        igual(SL1,SD1),igual(SL1,SD2).
33        ve_restricoes(e(Ln1,L)):- \+igual(Ln1, []), ve_repetidos(L).
34        % verifica se existem valores repetidos
35        ve_repetidos([]).
36        ve_repetidos([v(_,_,V)|T]):-
37        \+member(v(_,_,V),T), ve_repetidos(T).
38
39        %Função sucessor
40        sucessor(e([v(N,D,V)|R],E),e(R,[v(N,D,V)|E])):- member(V,D).
```

Os estados são apresentados por uma lista de variáveis. Cada um dos `v(p(X,Y), D, V)` representa uma variável, que representa uma posição do tabuleiro, com as respetivas coordenadas, domínio e valor na posição. O domínio restringe os valores que a posição pode tomar.

b) Resolva o problema com o algoritmo de backtracking.
Resultado Obtido:

```
 2 | 7 | 6
---+---+---
 9 | 5 | 1
---+---+---
 4 | 3 | 8
Fim
```

c) Resolva o problema modificando o algoritmo anterior para que faça verificação para a frente (forward checking).

```
back_f:- estado_inicial(E0), back(E0,A),sort(A, L),write('␣'),
    escreve(L).
```

d) Modifique o algoritmo anterior como entender de forma melhorar a complexidade (temporal e espacial).

Apesar de não termos implementado as alterações ao algoritmo, uma possibilidade para melhorar a complexidade do mesmo, seria aplicar uma heurística de ordenação de valores para escolher os valores mais promissores primeiro.

e) No relatório apresente os resultados para 4 exemplos diferentes:

```
| ?- p.
 2 | 7 | 6
---+---+---     Este estado e obtido a partir do estado inicial
 9 | 5 | 1      a zeros.
---+---+---
 4 | 3 | 8
Fim

| ?- p.
 8 | 1 | 6
---+---+---     Este estado e obtido com a posição 1 ja
 3 | 5 | 7      instanciada com o numero 8.
---+---+---
 4 | 9 | 2
Fim

| ?- p.
 4 | 3 | 8
---+---+---     Este estado e obtido com a posição 2 ja
 9 | 5 | 1      instanciada com o numero 3.
---+---+---
 2 | 7 | 6
Fim

| ?- p.
 4 | 9 | 2
---+---+---     Este estado e obtido a partir da posição 7 ja
 3 | 5 | 7      instanciada com o numero 8.
---+---+---
 8 | 1 | 6
Fim
```

2 Considere o problema do sudoku.

(a) Represente este problema como um problema de satisfação de restrições em prolog.

Em baixo encontra-se o estado inicial do próprio sudoku.

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

```
57              v(p(9, 8), [1,2,3,4,5,6,7,8,9], _),
58              v(p(9, 9), [1,2,3,4,5,6,7,8,9], _)],
59              [v(p(1, 2), [1,2,3,4,5,6,7,8,9], 1),
60              v(p(1, 6), [1,2,3,4,5,6,7,8,9], 8),
61              v(p(1, 8), [1,2,3,4,5,6,7,8,9], 7),
62              v(p(1, 9), [1,2,3,4,5,6,7,8,9], 3),
63              v(p(2, 4), [1,2,3,4,5,6,7,8,9], 5),
64              v(p(2, 6), [1,2,3,4,5,6,7,8,9], 9),
65              v(p(3, 1), [1,2,3,4,5,6,7,8,9], 7),
66              v(p(3, 7), [1,2,3,4,5,6,7,8,9], 9),
67              v(p(3, 9), [1,2,3,4,5,6,7,8,9], 4),
68              v(p(4, 6), [1,2,3,4,5,6,7,8,9], 4),
69              v(p(5, 5), [1,2,3,4,5,6,7,8,9], 3),
70              v(p(5, 6), [1,2,3,4,5,6,7,8,9], 5),
71              v(p(5, 8), [1,2,3,4,5,6,7,8,9], 1),
72              v(p(5, 9), [1,2,3,4,5,6,7,8,9], 8),
73              v(p(6, 1), [1,2,3,4,5,6,7,8,9], 8),
74              v(p(6, 4), [1,2,3,4,5,6,7,8,9], 9),
75              v(p(7, 4), [1,2,3,4,5,6,7,8,9], 7),
76              v(p(8, 1), [1,2,3,4,5,6,7,8,9], 2),
77              v(p(8, 2), [1,2,3,4,5,6,7,8,9], 6),
78              v(p(8, 5), [1,2,3,4,5,6,7,8,9], 4),
79              v(p(8, 8), [1,2,3,4,5,6,7,8,9], 3),
80              v(p(9, 3), [1,2,3,4,5,6,7,8,9], 5),
81              v(p(9, 6), [1,2,3,4,5,6,7,8,9], 3)])).
```

Cada um dos `v(p(X,Y), D, V)` representa uma variável, que representa uma posição do tabuleiro, com as respetivas coordenadas, domínio e valor na posição. O domínio restringe os valores que a posição pode tomar. Abaixo encontra-se um exemplo de representação de uma variável já instanciada com valor 3, na posição com coordenadas (9, 6).

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

De seguida encontram-se as restrições utilizadas, que consistem em impedir que sejam repetidos números na mesma linha, coluna ou quadrante.

```
1     verifica_restricao(E):-
2             verifica_linhas(E),
3             verifica_colunas(E),
4             verifica_quadranterantes(E).
5
6
7     verifica_linhas(e(_,[v(p(X,_), _, V)|R])):-
8             findall(V1,member(v(p(X,_),_,V1),R),L),
9             todos_diferentes([V|L]).
10
11    verifica_colunas(e(_,[v(p(_,Y), _, V)|R])):-
12            findall(V1,member(v(p(_,Y),_,V1),R),L),
13            todos_diferentes([V|L]).
14
15
16    verifica_quadranterantes(e(_,Afectado)) :-
17            verifica_quadrante(Afectado, 1, 1, 3, Quadrante1),
18            todos_diferentes(Quadrante1),
19            verifica_quadrante(Afectado, 1, 4, 6, Quadrante2),
20            todos_diferentes(Quadrante2),
21            verifica_quadrante(Afectado, 1, 7, 9, Quadrante3),
22            todos_diferentes(Quadrante3),
23            verifica_quadrante(Afectado, 4, 1, 3, Quadrante4),
```

```
            todos_diferentes(Quadrante4),
            verifica_quadrante(Afectado, 4, 4, 6, Quadrante5),
            todos_diferentes(Quadrante5),
            verifica_quadrante(Afectado, 4, 7, 9, Quadrante6),
            todos_diferentes(Quadrante6),
            verifica_quadrante(Afectado, 7, 1, 3, Quadrante7),
            todos_diferentes(Quadrante7),
            verifica_quadrante(Afectado, 7, 4, 6, Quadrante8),
            todos_diferentes(Quadrante8),
            verifica_quadrante(Afectado, 7, 7, 9, Quadrante9),
            todos_diferentes(Quadrante9).


    verifica_quadrante(L, X, Y, Y2, Q) :-
            Y = Y2, X1 is X+2,
            adiciona_quadrante(L, X, Y, X1, Q).

    verifica_quadrante(L, X, Y, Y2, Q) :-
            Y < Y2, Y1 is Y+1,
            X1 is X+2,
            adiciona_quadrante(L, X, Y, X1, L1),
            append(L1, L2, Q),
            verifica_quadrante(L, X, Y1, Y2, L2).

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

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

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

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

    todos_diferentes([]).
    todos_diferentes([X|R]):-
            \+ member(X,R), todos_diferentes(R).
```

Por último, o predicado sucessor.

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

(b) Resolva o problema com o algoritmo de backtracking.

Algoritmo de Backtracking:

```
    back(e([],A),A).
    back(E,Solucao):-
            sucessor(E,E1),
            verifica_restricao(E1),
            back(E1,Solucao).
```

Resultado:

```
 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
```

(c) Resolva o problema modificando o algoritmo anterior para que faça verificação para a frente (forward checking).

Algoritmo para Forward checking:

```
forward(e([],A),A).
forward(E,Solucao):-
        sucessor(E,E1),
        verifica_restricao(E1),
        forCheck(E1, E2),
        forward(E2, Solucao).

forCheck(e(Lni,[v(N,D,V)|Li]), e(Lnii, [v(N,D,V)|Li])) :- corta(V,
    Lni,Lnii).
```

É possível verificar que é utilizado um predicado `corta/3` no Forward checking. Este predicado é utilizado com o objetivo de remover do domínio de variáveis ainda por instanciar, valores que estas já não podem tomar.

```
corta(_, [], []).
corta(V, [v(Ni, D, _)|Li], [v(Nj, D1, _)|Lii]):-
        restricoes(Ni, Nj),
        delete(D, V, D1),
        corta(V, Li, Lii).
```

Apesar de várias tentativas, não foi possível obter um resultado válido ao utilizar o algoritmo Forward Check. Cada execução terminava com um no, e, mesmo após diversas tentativas de resolução, não conseguimos identificar e corrigir o problema.

(d) Modifique o algoritmo anterior como entender de forma melhorar a complexidade (temporal e espacial).

Apesar de não termos implementado as alterações ao algoritmos, uma proposta para melhorar a complexidade do algoritmos , seria a utilização de heurísticas avançadas estas heurísticas podem ser desenvolvidas com base no conhecimento do domínio do problema e podem orientar a procura de uma forma mais eficiente.