

# Inteligência Artificial

Trabalho 2015/2016

05/04/16



UNIVERSIDADE  
DE ÉVORA

2	7	6	→15	
9	5	1	→15	
4	3	8	→15	
↙15	↓15	↓15	↓15	↘15

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

Professora:  
Irene Pimenta Rodrigues

Realizado por:  
João Calhau - 31621  
José Pimenta - 31677

# Índice

Introdução.....	Pag 3
Exercicio 1.....	Pag 4
Exercicio 1.....	Pag 7
Conclusão.....	Pag 9

# Introdução

Este trabalho enquadra-se na disciplina de Inteligência Artificial e vamos abordar pesquisas com constraints para resolução de um exercício definido pela professora, em que temos de resolver o quadrado mágico e o sudoku.

Iremos assim tentar dar o nosso melhor, e iremos tentar utilizar os métodos que achemos mais correctos ou propícios à boa evolução do trabalho e que no final se concretize o que nos é pedido.

## Exercicio 1:

Para representar o problema através do uso de variáveis como um problema de satisfação de restrições, estabelecemos que uma variável tem 3 elementos, o 1º é a posição  $p(X,Y)$  do tabuleiro, o 2º é o domínio e o 3º é o valor dessa posição.

Será do tipo (para 3x3):

estado\_inicial(E).

```
| ?- estado_inicial(E).  
E = e([v(p(1,1),[1,2,3,4,5,6,7,8,9],_),v(p(2,1),[1,2,3,4,5,6,7,8,9],_),v(p(3,1),[1,2,3,4,5,6,7,8,9],_),v(p(1,2),[1,2,3,4,5,6,7,8,9],_),v(p(2,2),[1,2,3,4,5,6,7,8,9],_),v(p(3,2),[1,2,3,4,5,6,7,8,9],_),v(p(1,3),[1,2,3,4,5,6,7,8,9],_),v(p(2,3),[1,2,3,4,5,6,7,8,9],_),v(p(3,3),[1,2,3,4,5,6,7,8,9],_)],[]) ? |
```

Neste caso o domínio vai de 1 até 9, pois é  $3 \times 3 = 9$ , mas num 4x4 vai até 16, e num 5x5 vai até 25.

Para as restrições, temos a função `ve_restricoes(E)`, que chama `ver_tudo`, `ver_linhas`, `ver_colunas` e 2 funções de diagonais.

```
%Restricoes
ve_restricoes(E):-
    tamanho_tabuleiro(T),
    ver_tudo(E),
    ver_linhas(E,L1,T1),
    ver_colunas(E,L2,T2),
    ver_diagonal_dir(E,L3,T3),
    ver_diagonal_esq(E, L4, T4),

    conta(K),

    (=(T, T1)
     -> (=(K, L1)
        -> true
        ; fail)
     ; true),

    (=(T, T2)
     -> (=(K, L2)
        -> true
        ; fail)
     ; true),

    (=(T, T3)
     -> (=(K, L3)
        -> true
        ; fail)
     ; true),

    (=(T, T4)
     -> (=(K, L4)
        -> true
        ; fail)
     ; true).
```

```
ver_tudo(e(Nafect,[v(p(_,_), D, V)|R])):-
    findall(V1,member(v(p(_,_),_,V1),R),L),
    todos_diff([V|L]).

ver_linhas(e(Nafect,[v(p(X,Y), D, V)|R]),S, T):-
    findall(V1,member(v(p(X,_),_,V1),R),L),
    todos_diff([V|L]), length([V|L],T),
    sum_list([V|L],S).

ver_colunas(e(Nafect,[v(p(X,Y), D, V)|R]),S, T):-
    findall(V1,member(v(p(_,Y),_,V1),R),L),
    todos_diff([V|L]), length([V|L],T),
    sum_list([V|L],S).
```

```

ver_diagonal_dir(e(Nafect,[v(p(X,Y), D, V)|R]),S, T):-
    tamanho_tabuleiro(Tb),
    Ta is Tb * Tb,
    findall((K1,K2,V1),member(v(p(K1,K2),_,V1),R),L),
    length([(X,Y,V)|L],Tc),

    (=(Tc,Ta)
    ->
    (verify_dir([(X,Y,V)|L], Result), diag(Diag),
    sum_list(Diag,S), length(Diag,T))
    ;
    true).

ver_diagonal_esq(e(Nafect,[v(p(X,Y), D, V)|R]),S, T):-
    tamanho_tabuleiro(Tb),
    Ta is Tb * Tb,
    findall((K1,K2,V1),member(v(p(K1,K2),_,V1),R),L),
    length([(X,Y,V)|L],Tc),
    (=(Tc,Ta)
    ->
    (verify_esq([(X,Y,V)|L], Result), diag2(Diag),
    sum_list(Diag,S), length(Diag,T))
    ;
    true).

```

O operador sucessor é o mesmo que foi utilizado no problema das 8 rainhas:

```

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

```

O sucessor indica que valor pode ser atribuído a qualquer variável não instanciada, desde que não entre em conflito com as restrições definidas, instanciando uma variável e colocando-a na lista Afect.

O problema resolve-se bem com problema de backtracking, tendo em seguida, resultados para 3x3, 4x4. (Pesquisa em Backtracking)

```

| ?- p.
2 . 9 . 4      8 . 1 . 6
7 . 5 . 3      3 . 5 . 7
6 . 1 . 8      4 . 9 . 2

true ?
(47 ms) yes

```

= 15

```

| ?- p.
1 . 12 . 13 . 8
2 . 14 . 7 . 11
15 . 3 . 10 . 6
16 . 5 . 4 . 9

1 . 13 . 12 . 8
2 . 14 . 7 . 11
15 . 3 . 10 . 6
16 . 4 . 5 . 9

true ?
(93641 ms) yes

```

=34

## Exercicio 2:

Tal como no exercício do quadrado mágico, representamos o problema com o uso de uma variável com 3 elementos, elementos estes que são a coordenada espacial, o domínio e o valor do elemento. Neste caso o valor de X e Y vão até 9 visto ser um tabuleiro 9x9 e o domínio até 9 visto o sudoku tratar-se de um quadrado 9x9, com 9 quadrantes 3x3.

Para vermos restrições baseamo-nos nas restrições do exercício do quadrado mágico (ver\_linhas e ver\_colunas) e retiramos a parte em que verificávamos a soma das linhas/colunas. Fizemos ainda uma restrição auxiliar (ver\_quadrantes) que verifica se todos os 9 quadrantes 3x3 tem valores diferentes entre eles.

```
ver_quad(L, X, Y, Y2, Q) :-
    Y = Y2, X1 is X+2,
    add_quad(L, X, Y, X1, Q).

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

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

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

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

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

ve_restricoes(E) :-
    ver_linhas(E),
    ver_colunas(E),
    ver_quadrantes(E).

ver_linhas(e(Naffect, [v(p(X,Y), D, V)|R])) :-
    findall(V1, member(v(p(X, _), _, V1), R), L),
    todos_diff([V|L]).

ver_colunas(e(Naffect, [v(p(X,Y), D, V)|R])) :-
    findall(V1, member(v(p(_, Y), _, V1), R), L),
    todos_diff([V|L]).

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

Como os valores de cada quadrante só podem chegar a 9, a resolução não demora muito tempo.

```

| ?- p.
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

```

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

Resultado na consola do GNU prolog.

Matriz com valores iniciais.

```

| ?- p.
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

```

Comparação com valores iniciais (em posições correctas)

Resultado é apresentado rapidamente, e por termos tido algumas complicações na implementação no forward check, este não foi testado no forward checking.



# Conclusão

Após a realização deste trabalho, ficámos a conhecer melhor como funcionam algoritmos de pesquisa com constraints. Aplicámos conhecimentos adquiridos de modo a tentar resolver os problemas propostos pela professora, e conseguimos resolver minimamente os problemas propostos. Conseguimos resolver o Quadrado mágico para 3x3 e 4x4 em bom tempo, e conseguimos resolver o Problema do Sudoku. Ainda no caso do quadrado mágico, no problema 5x5, demora muito tempo, e por isso o algoritmo de backtracking não é aconselhado.

Tivemos dificuldades em implementar o algoritmo de forward check, e pelo que o algoritmo de backtracking funcionou para os casos já referidos e por questões de tempo, tentámos implementar, mas não ficou pronto para funcionar neste trabalho.