

## **Agente Solucionador para o Problema das N- Rainhas Utilizando Forward Checking com *most-constrained-variable* e *least- constraining-value***

Natanael Maia  
COPPE/UFRJ

# N-Rainhas

- Posicionar  $N$  rainhas num tabuleiro ( $N \times N$ ) de maneira que nenhuma rainha ataque outra rainha
- O problema original, conhecido como 8-rainhas possui 92 soluções, sendo apenas 12 distintas

# N-Rainhas

N	Total de Soluções	Soluções Distintas
1	1	1
2	0	0
3	0	0
4	2	1
5	10	2
6	4	1
7	40	6
<b>8</b>	<b>92</b>	<b>12</b>
9	352	46
10	724	92
11	2,680	341
12	14,200	1,787
13	73,712	9,233
14	365,596	45,752
15	2,279,184	285,053
16	14,772,512	1,846,955
17	95,815,104	11,977,939
18	666,090,624	83,263,591
19	4,968,057,848	621,012,754
20	39,029,188,884	4,878,666,808
21	314,666,222,712	39,333,324,973
22	2,691,008,701,644	336,376,244,042
23	24,233,937,684,440	3,029,242,658,210

- Tipos de Formulação:
  - Formulação incremental
  - Formulação de estado completo

# Formulação Incremental

- Utiliza *Forward Checking*
  - A cada rainha posicionada, os valores conflitantes são retirados dos domínios das outras rainhas não posicionadas. Se não for possível posicionar uma rainha (domínio vazio), realiza *backtracking*
- Heurísticas:
  - *Most-constrained-variable*: escolhe a variável com o domínio mais restrito
  - *Least-constraining-value*: escolhe o valor que restringe menos os domínios das outras variáveis

# *Forward Checking*

```
public Action[] search(Problem problem) {  
    Node node = new Node(problem.getInitialState());  
    nodes.insertElementsFront(expand(node));  
    while (!nodes.isEmpty()) {  
        node = (Node) nodes.removeFront();  
        incrementSteps();  
        if (problem.goalTest(node.getState())) {  
            problem.setInitialState(node.getState());  
            return node.getActionSequence();  
        }  
        nodes.insertElementsFront(expand(node));  
    }  
    return null;  
}
```

# Forward Checking

- Soluções:

*most-constrained-variable*

The screenshot shows the NQueen application interface. The main window is titled "NQueen". It features a chessboard labeled "Tabuleiro" with 8 queens placed on it. To the right of the chessboard is a list of solutions labeled "Soluções". The list contains 92 solutions, each represented as a sequence of 8 numbers. The bottom section, labeled "Configurações", includes a spinner for "N" set to 8, a dropdown for "Tipo da Formulação" set to "Incremental", and fields for "Passos" (1796) and "Resultados" (92). At the bottom are three buttons: "Limpar tudo", "Próxima solução", and "Todas as soluções".

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

**Configurações**

N: 8

Tipo da Formulação: Incremental

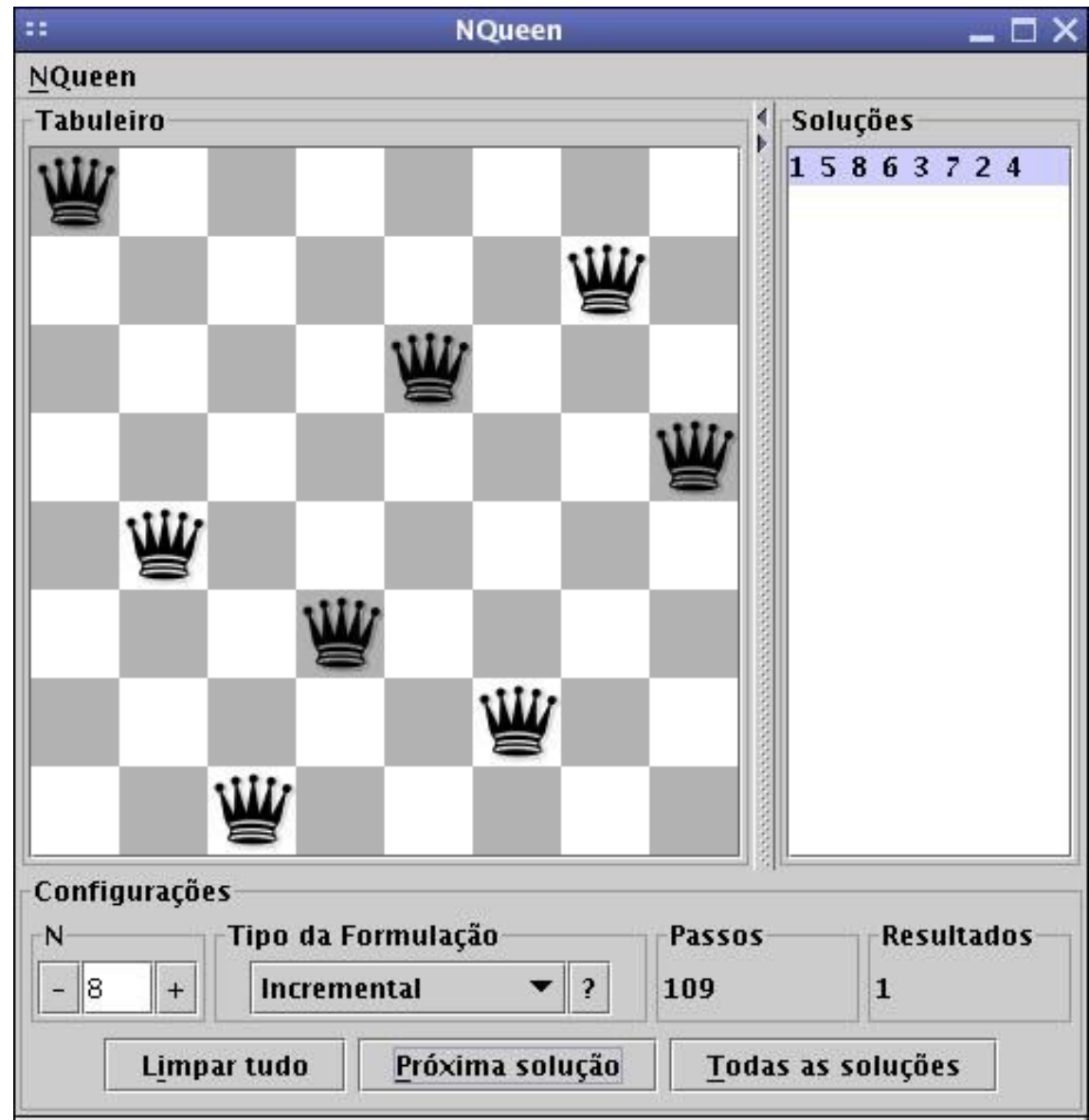
Passos: 1796

Resultados: 92

Botões: Limpar tudo, Próxima solução, Todas as soluções

# Forward Checking

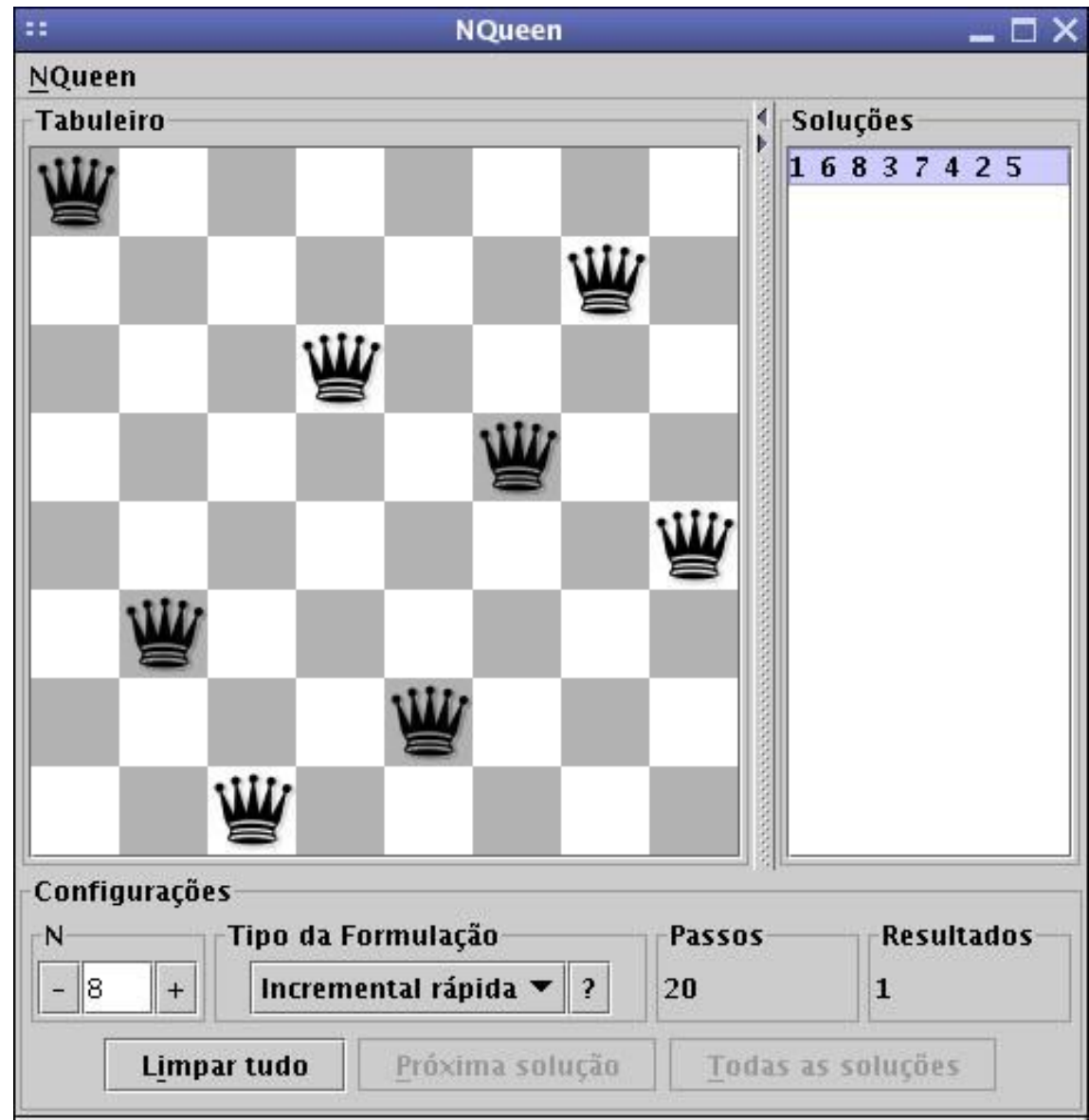
- Primeira solução:  
*most-constrained-variable*
- 109 passos





# Forward Checking

- Primeira solução:  
*most-constrained-variable*  
*least-constraining-value*
- 20 passos



# Comparativo

- Cálculo do número de passos e tempo gasto para encontrar uma solução para cada  $N$  variando de 4 até 80
  - Utilizando somente *most-constrained-variable*:
    - 310.121 passos em aprox. 1m15s
  - Utilizando *most-constrained-variable* e *least-constraining-value*:
    - 19.401 passos em aprox. 4s
    - Atingiu  $N=155$  em 30 minutos

# Download

- <http://www.cos.ufrj.br/~ntmaia/nqueen.zip>