INF623

2024/1

UFV

Inteligência Artificial

A9: Problemas de satisfação de restrição I

IA na mídia

03/04/2024

Google Vlogger
Modelo desenvolvido pela Google para animação de imagens com base em texto e imagem.

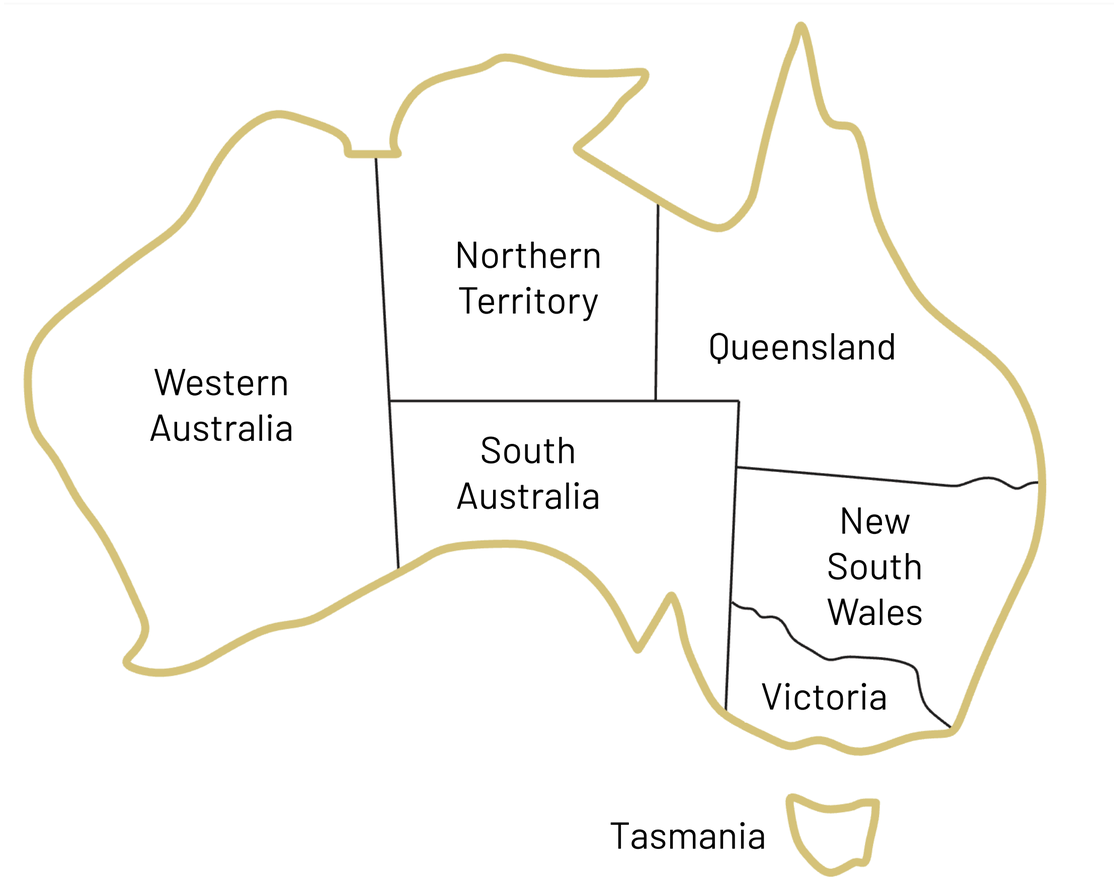
https://enriccorona.github.io/vlogger/

Plano de aula

- Problemas de satisfação de restrições
- Exemplos
- Tipos de restrições
- Grafo de restrições
- Busca com retrocesso

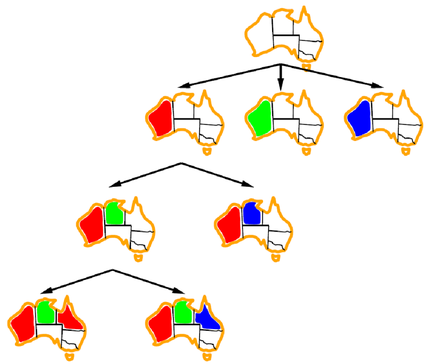
Exemplo 1: coloração de mapas

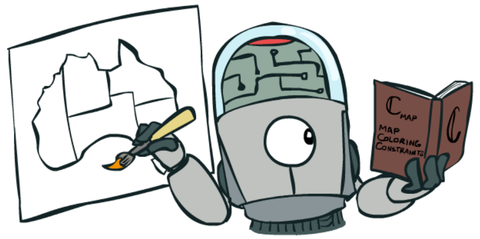
Considere o problema de colorir o mapa de um país com um número restrito de cores de tal forma que nenhum de seus estados adjacentes possuam a mesma cor.



Agentes racionais para jogos

Para resolver problemas desse tipo, chamados de **Problemas de Satisfação de Restrição (PSRs)**, um agente assume que o mundo é representado por um **espaço de estados fatorados** e que objetivo é encontrar uma solução que satisfaça um **conjunto de restrições**.





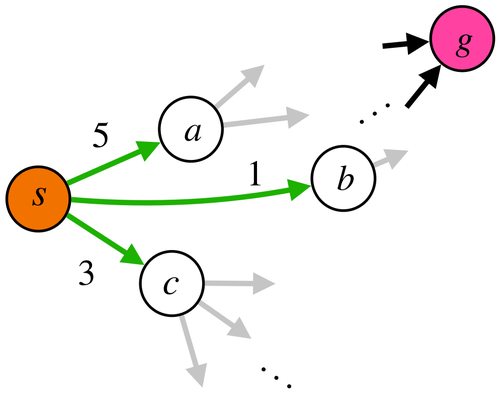
- **Estados** são **fatorados** em um conjunto de variáveis;
- Um **conjunto de restrições** especifica combinações possíveis de variáveis;
- Uma **solução** é uma atribuição de valores para todas as variáveis que satisfaça o conjunto de restrições

Problemas de satisfação de restrição

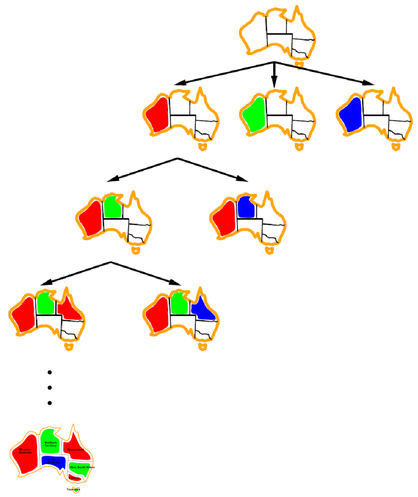
Um **problema de satisfação de restrição** pode ser definito pela tupla (X, D, C), onde:

- X é um conjunto de **variáveis** $\{X_1, \ldots, X_n\}$
- D é um conjunto de **domínios** $\{D_1, \ldots, D_n\}$
  - Um domínio D_i é um conjunto de valores possíveis $\{v_1, \ldots, v_k\}$ para X_i (e.g., variáveis booleanas – $\{V, F\}$)
  - Diferentes variáveis podem ter diferentes domínios com tamanhos diferentes
- C é um conjunto de **restrições** que especificam combinações de valores possíveis
  - Cada restrição C_j consiste em um par (Escopo, Relação), onde:
    - *Escopo* é uma tupla de variáveis que participam na restrição (e.g., (X_1, X_2));
    - *Relação* pode ser expressa:
      - *Explicitamente* – pelo conjunto de todas as tuplas de valores que satisfazem a restrição (e.g.,);
      - *Implicitamente* – por uma função que retorna se uma tupla faz parte da relação
- Uma solução é uma atribuição $\{X_i = v_i, X_j = v_j, \ldots\}$ que seja **completa** e **consistente**
  - **Atribuição completa:** cada variável tem um valor atribuído
  - **Atribuição consistente:** não viola nenhuma restrição

Busca no espaço de estados vs. PSRs



PSRs são geralmente NP-completos, mas existem algumas sub-classes de problemas que podem ser resolvidos de maneira eficiente



- Objetivo é encontrar um caminho entre s e g
- Busca no espaço de **estados**
- **Função de ações A:** dado um estado, gerar próximos estados
- Função heurística h específica do problema guia a busca em direção da solução

- Objetivo é encontrar g (uma atribuição de valores que valores que satisfação o conjunto de restrições)
- Busca no espaço de **estados fatorados** (atribuições)
- **Função de ações A:** atribuir um valor a uma variável
- A busca é guiada pela própria estrutura dos estados (abordagem mais geral)

UFV

Exemplo 1: coloração de mapas

- **Variáveis:** $X = \{\text{WA}, \text{NT}, \text{Q}, \text{NSW}, \text{V}, \text{SA}, \text{T}\}$
- **Domínios:** $D = \{\text{Vermelho}, \text{Verde}, \text{Azul}\}$
- **Restrições:** Regiões adjacentes devem ter cores diferentes
  - Explicitamente:

    $(\text{WA}, \text{NT}) \in \{(\text{Vermelho}, \text{Verde}), (\text{Vermelho}, \text{Azul}), \ldots\}$

    $(\text{WA}, \text{SA}) \in \{(\text{Vermelho}, \text{Verde}), (\text{Vermelho}, \text{Azul}), \ldots\}$

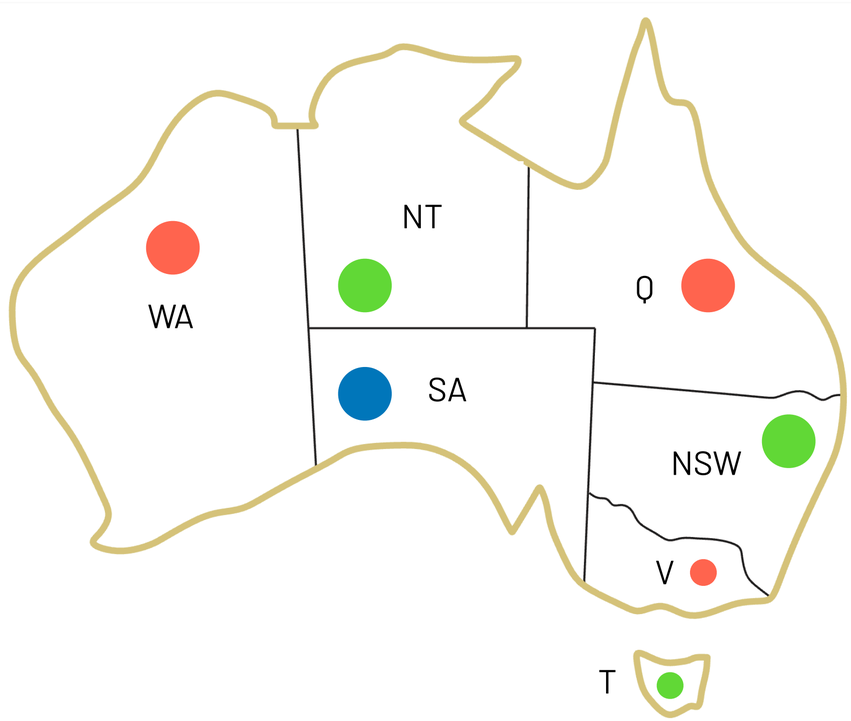
    $\ldots$
  - Implicitamente:

    $C = \{\text{WA} \neq \text{NT}, \text{WA} \neq \text{SA}, \text{NT} \neq \text{Q}, \text{NT} \neq \text{SA},$

    $\text{Q} \neq \text{SA}, \text{Q} \neq \text{NSW}, \text{NSW} \neq \text{SA}, \text{NSW} \neq \text{V}, \text{V} \neq \text{SA}\}$
- **Exemplo de solução**

  $\{\text{WA} = \text{Vermelho}, \text{NT} = \text{Verde}, \text{Q} = \text{Vermelho}, \text{NSW} = \text{Verde},$

  $\text{V} = \text{Vermelho}, \text{SA} = \text{Azul}, \text{T} = \text{Verde}\}$



Exemplo 2: Sudoku

- **Variáveis:** Uma variável $X_{i,j}$ para cada célula da matriz.
- **Domínios:** $D = \{1,2,3,4,5,6,7,8,9\}$
- **Restrições (Implicitamente):**
  - **Linhas**
    - Todas as 9 variáveis diferentes para linha 1 $(\{X_{1,1}, X_{1,2}, \ldots, X_{1,9}\}, \neq)$
    - Todas as 9 variáveis diferentes para linha 2 $(\{X_{2,1}, X_{2,2}, \ldots, X_{2,9}\}, \neq)$
    - ...
  - **Colunas**
    - Todas as 9 variáveis diferentes para coluna 1 $(\{X_{1,1}, X_{2,1}, \ldots, X_{9,1}\}, \neq)$
    - Todas as 9 variáveis diferentes para coluna 2 $(\{X_{1,2}, X_{2,2}, \ldots, X_{9,2}\}, \neq)$
    - ...
  - **Quadrantes**
    - Todas as 9 variáveis diferentes do 1º quadrante: $(\{X_{1,1}, X_{1,2}, X_{1,3}, X_{2,1}, X_{2,2}, X_{2,3}, X_{3,1}, X_{3,2}, X_{3,3}\}, \neq)$
    - Todas as 9 variáveis diferentes do 2º quadrante: $(\{X_{1,4}, X_{1,5}, X_{1,6}, X_{2,4}, X_{2,5}, X_{2,6}, X_{3,4}, X_{3,5}, X_{3,6}\}, \neq)$
    - ...

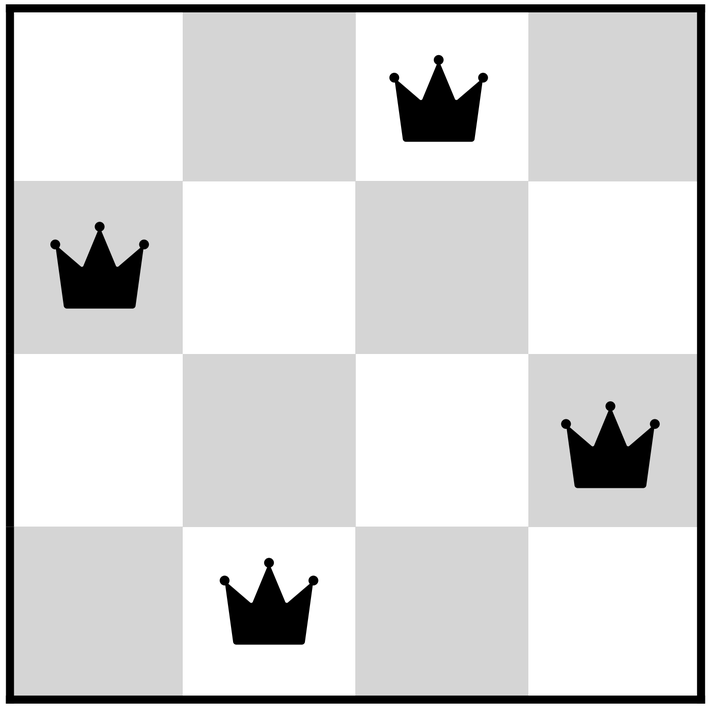
					8			4
	8	4		1	6			
			5			1		
1		3	8			9		
6		8				4		3
		2			9	5		1
		7			2			
			7	8		2	6	
2			3					

UFV

Exercício: 4-rainhas

Defina o problema das 4-rainhas como um PSR.

- **Variáveis:** _____
- **Domínios:** _____
- **Restrições:** _____
- **Exemplo de solução:** _____



Exercício: 4-rainhas

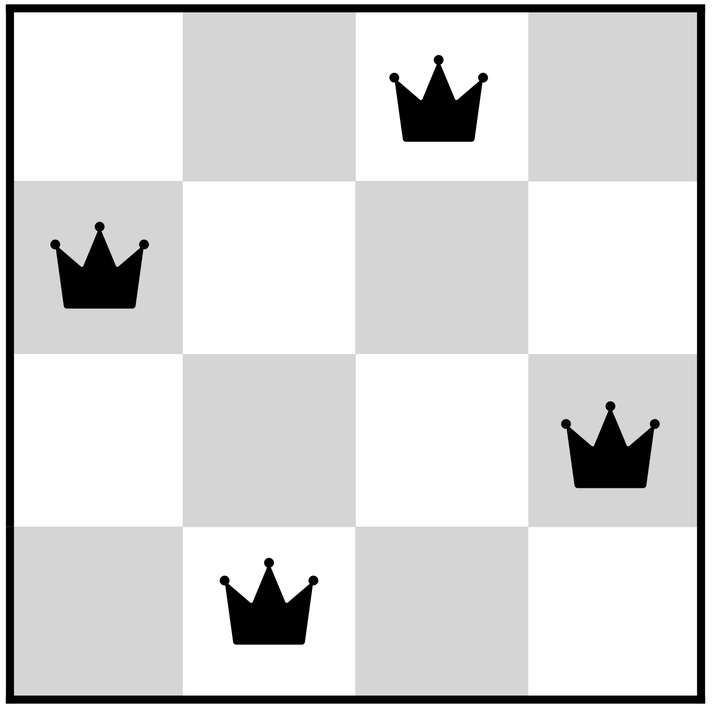
Formalização 1:

- **Variáveis:** Uma variável $X_{i,j}$ para cada posição do tabuleiro.
- **Domínios:** $D = \{0,1\}$
- **Restrições:**

$$\forall i,j,k\ (X_{ij}, X_{ik}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k\ (X_{ij}, X_{kj}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k\ (X_{ij}, X_{i+k,j+k}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k\ (X_{ij}, X_{i+k,j-k}) \in \{(0,0),(0,1),(1,0)\}$$
$$\sum X_{ij} = N$$

- **Exemplo de solução:**

$$\{X_{1,1} = 0, X_{1,2} = 0, X_{1,3} = 1, X_{1,4} = 0$$
$$X_{2,1} = 1, X_{2,2} = 0, X_{2,3} = 1, X_{2,4} = 0$$
$$X_{3,1} = 0, X_{3,2} = 0, X_{3,3} = 1, X_{3,4} = 1$$
$$X_{4,1} = 0, X_{4,2} = 1, X_{4,3} = 1, X_{4,4} = 0\}$$



Exercício: 4-rainhas

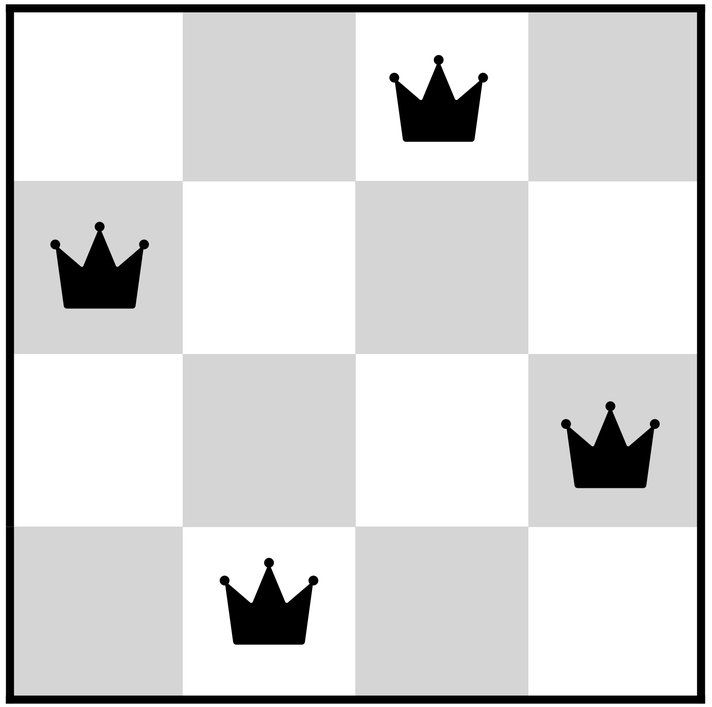
Formalização 2:

- **Variáveis:** Uma variável Q_i para cada coluna do tabuleiro.
- **Domínios:** $D = \{1,2,3,4\}$
- **Restrições:**

  Implicitamente: $\forall_{i,j}$ **nao-se-atacam**(Q_i, Q_j)

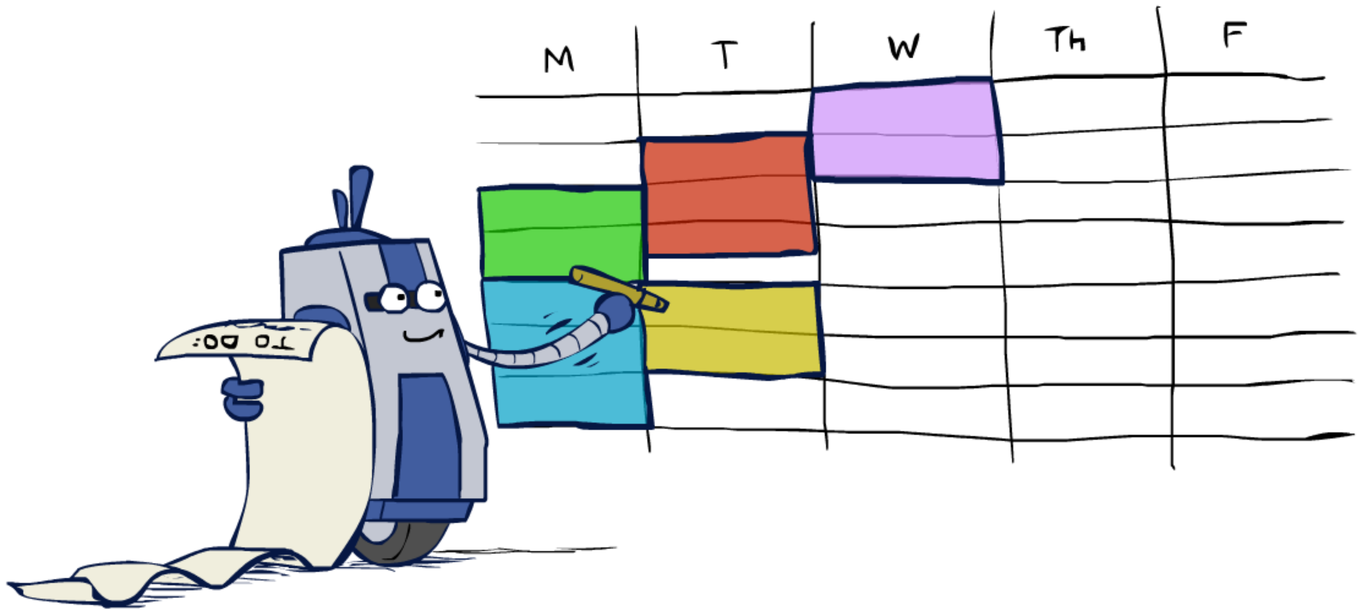
  Explicitamente: $(Q_1, Q_2) \in \{(1,3), (1,4), \ldots\}$
  $\ldots$
- **Exemplo de solução:**

$$\{Q_1 = 2, X_2 = 4, X_3 = 1, X_4 = 3\}$$



Aplicações

- Agendamento: quando podemos nos encontrar?
- Alocar horários: que aulas são oferecidas, quando e onde?
- Atribuição: quem ensina qual aula?
- Configuração de hardware
- Agendamento de transportes
- Automação de fábricas
- Layout de circuito
- Diagnóstico de erro
- ... muito mais!



Tipos de restrições

- **Unárias**: restringem uma variável

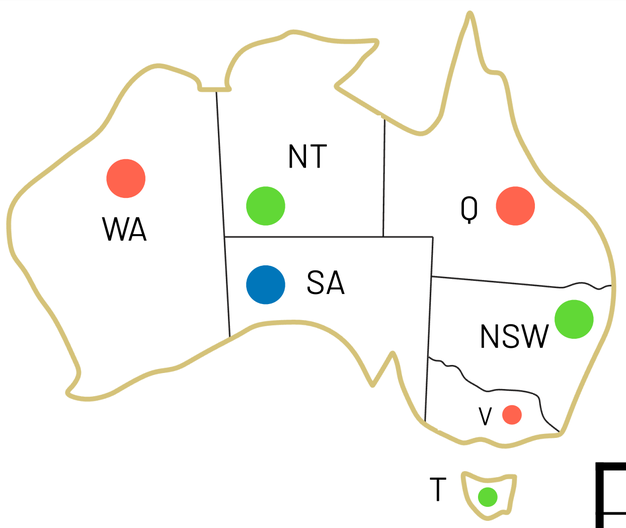
  Exemplo: WA não pode ser colorido de Verde

- **Binárias**: restringem duas variáveis

  Exemplo: estados adjacentes não podem ter a mesma cor

- **Globais**: restringem mais de duas variáveis

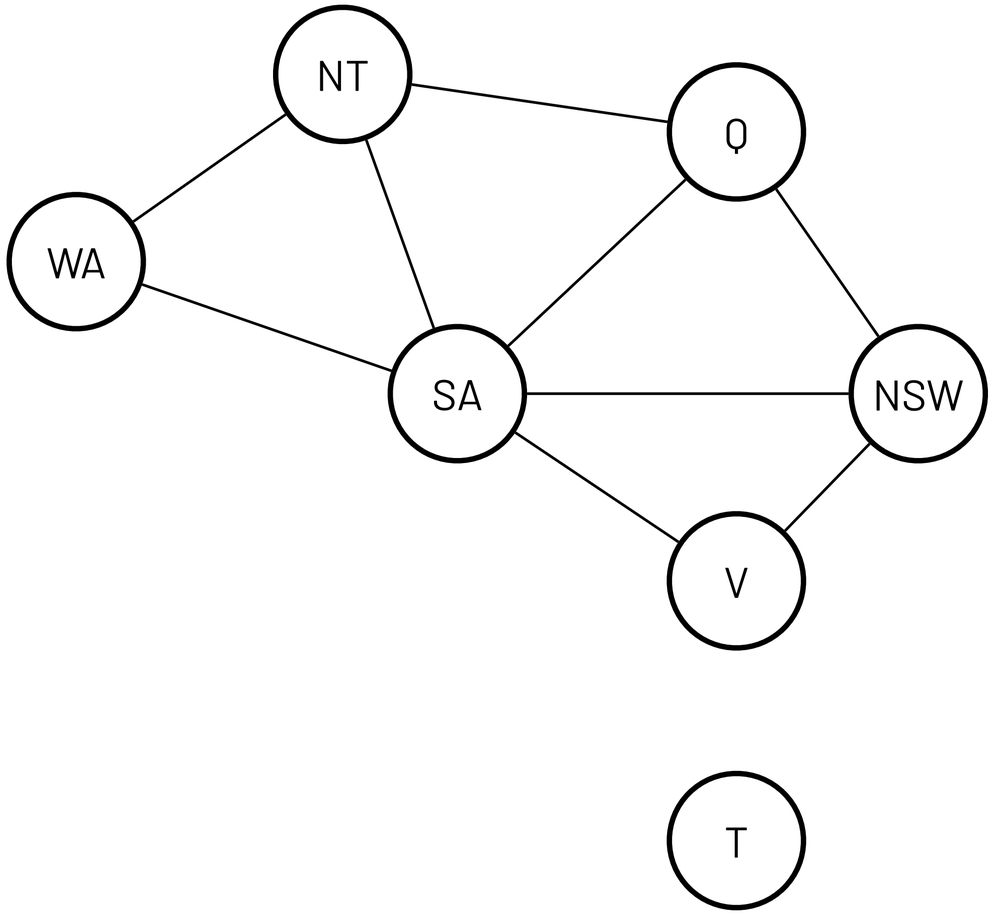
  Exemplo: todas as células de uma linha do Sudoku



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

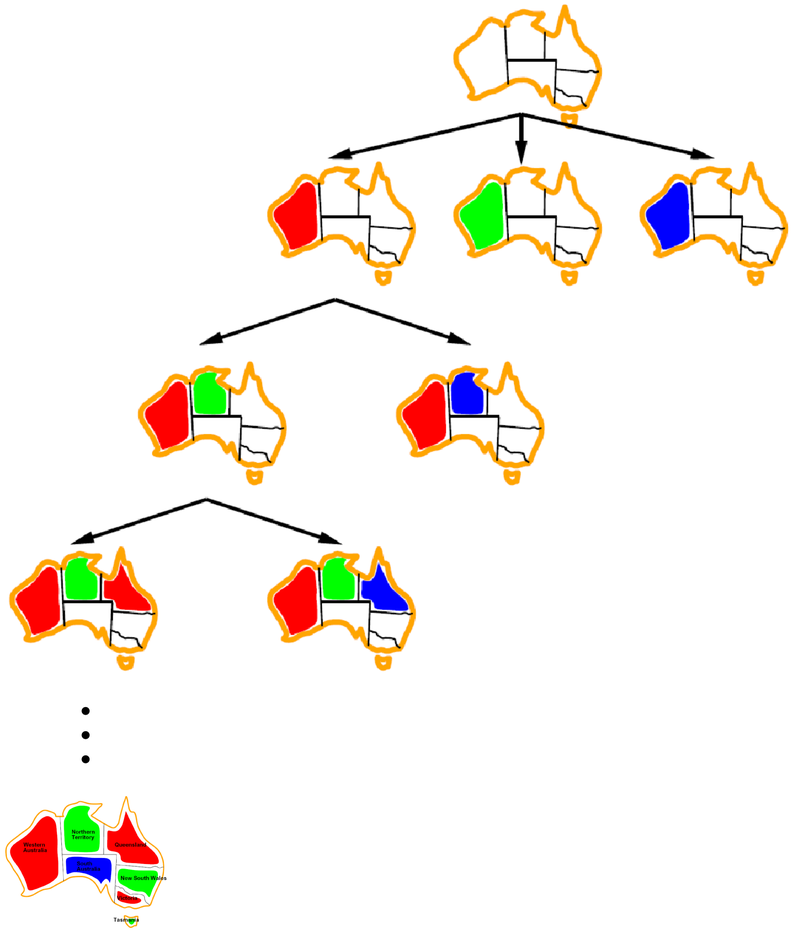
Grafo de restrições

- **Vértices** são variáveis
- **Arestas** representam restrições entre variáveis
  - Não especificam o tipo de restrição, mas apenas que ela existe
- Algoritmos de PSRs usam a estrutura do grafo de restrições para guiar a busca.
  - Exemplo: Tasmania é um problema independente!



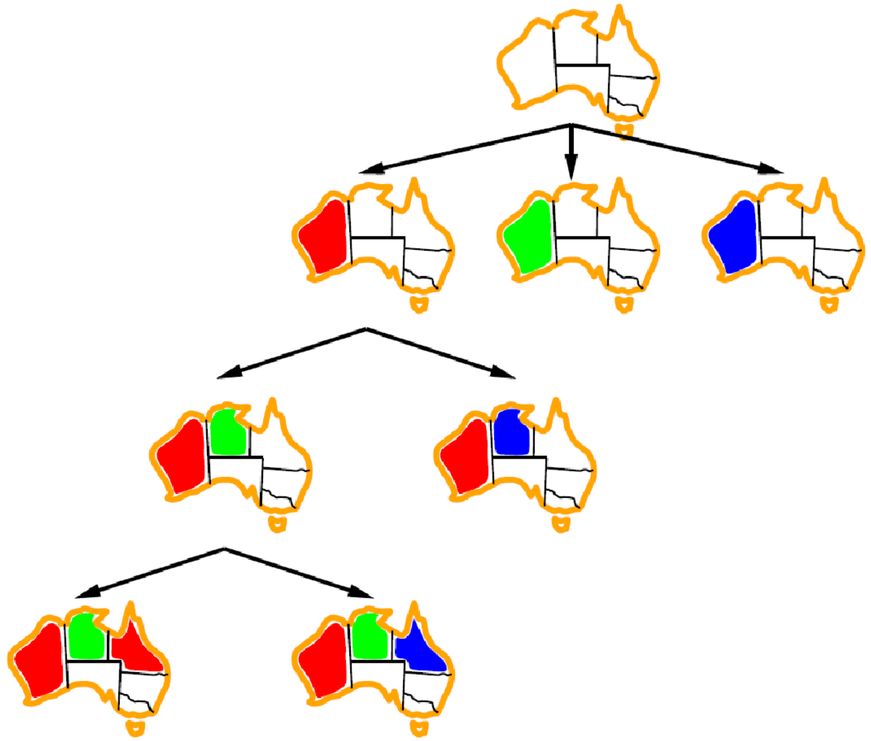
Resolvendo PSRs

- Formulação de busca para PSRs:
  - Estados são definidos por **atribuições parciais:**
    - **Estado Inicial:** atribuição vazia { }
    - **Função de Ação:** atribuir um valor para uma variável aberta
    - **Estado final:** a atribuição atual é **completa** a **consistente**
- Como seria o desempenho da BFS?
  - Todas as soluções estão nas folhas! A BFS exploraria todos os níveis sistematicamente antes de encontrar uma solução



Busca com retrocesso para PSRs: ideia geral

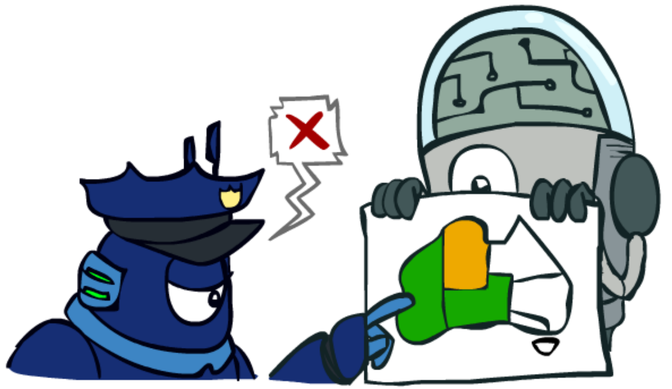
- **Busca com retrocesso** (*backtracking search*) é o algoritmo sem informação básico para resolver PSRs
- Modificar **busca em profundidade (DFS)** adicionando duas ideias:
  - **Ideia 1: Modificar uma única variável por vez**
    - Atribuição de variáveis é comutativa
    - Definir uma ordem para as variáveis
  - **Ideia 2: Verifique restrições a cada expansão**
    - Se chegarmos em um estado que não satisfaz as restrições, podemos retroceder!
    - Verificar restrições pode envolver computação não-constante
- Esse algoritmo consegue resolver o N-rainhas com $N \approx 25$



Busca com retrocesso para PSRs

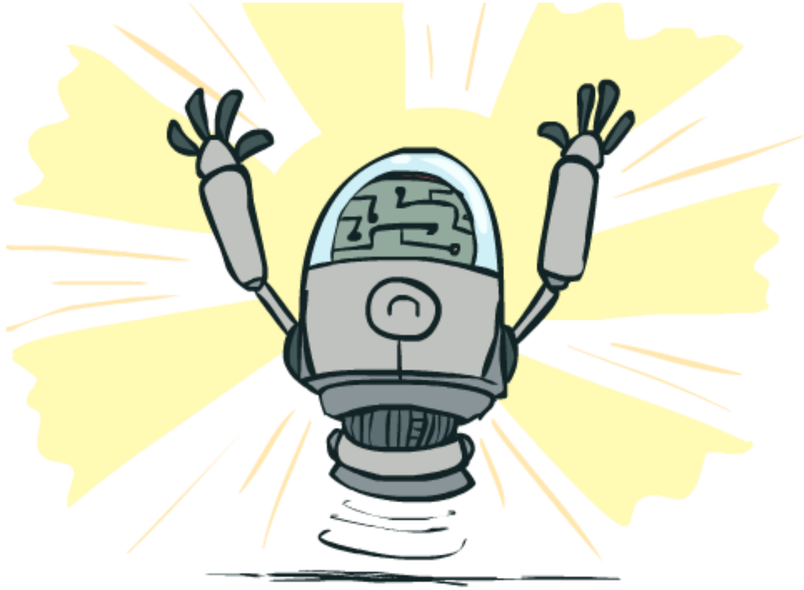
```
def busca-retrocesso(X, D, C):
1.      return retrocesso-recursivo({}, X, D, C)

def retrocesso-recursivo(assign, X, D, C):
1.      if is-complete(assign):
2.          return assign
3.      var = select-unsassigned-variable(X, assign)
4.      for value in sort(var, assign, X, D): # Ideia 1: ordenar variáveis
5.          if is-consistent(assign, C):      # Ideia 2: verificar restrições
6.              assign[var] = value
7.              result = busca-retrocesso(assign, X, D, C)
8.              if result != None:
9.                  return result
10.             else:
11.                 del assign[var]
12.     return None
```



Melhorias para a busca com retrocesso

- Novas ideias para guiar a busca em retrocesso
- **Ordenação de variáveis e valores:**
  - Quais variáveis devemos olhar primeiro?
  - Em que ordem os valores devem ser avaliados?
- **Inferência em PSRs:** podemos identificar falhar antes?
- **Estrutura do problema:** podemos explorar a estrutura dos problemas?



Próxima aula

A10: Problemas de satisfação de restrição II

Ordenação de variáveis e valores, inferência em PSRs e estruturas de problemas