

Inteligência Computacional

Problema de Satisfação de Restrição (PSR)

Prof. Fabio Augusto Faria

Material adaptado de livro

“Inteligência Artificial, S. Russell e P. Norving”

2º semestre 2021



Introdução

- Aulas anteriores exploram a ideia que problemas podem ser resolvidos, buscando-se uma solução em **espaço de estados**:
 - Heurísticas específicas de domínio
 - Testando se é objetivo
- Agora utiliza-se **representação fatorada** para cada estado:
 - Conjunto de **variáveis**, cada qual com um valor;
 - O problema será resolvido quando todas variáveis tiverem um valor que **satisfaça todas as restrições** sobre a variável alvo.

Problema de Satisfação de Restrição

- Componentes em PSR:
 - X: variáveis;
 - D: domínio {valores possíveis para cada variável X_i };
 - C: restrições que especificam combinações de valores possíveis;
 - C_i : **<escopo,rel>**
 - **escopo** é uma tupla de variáveis que participam da restrição;
 - **rel** é uma relação que define os valores que essas variáveis podem assumir.

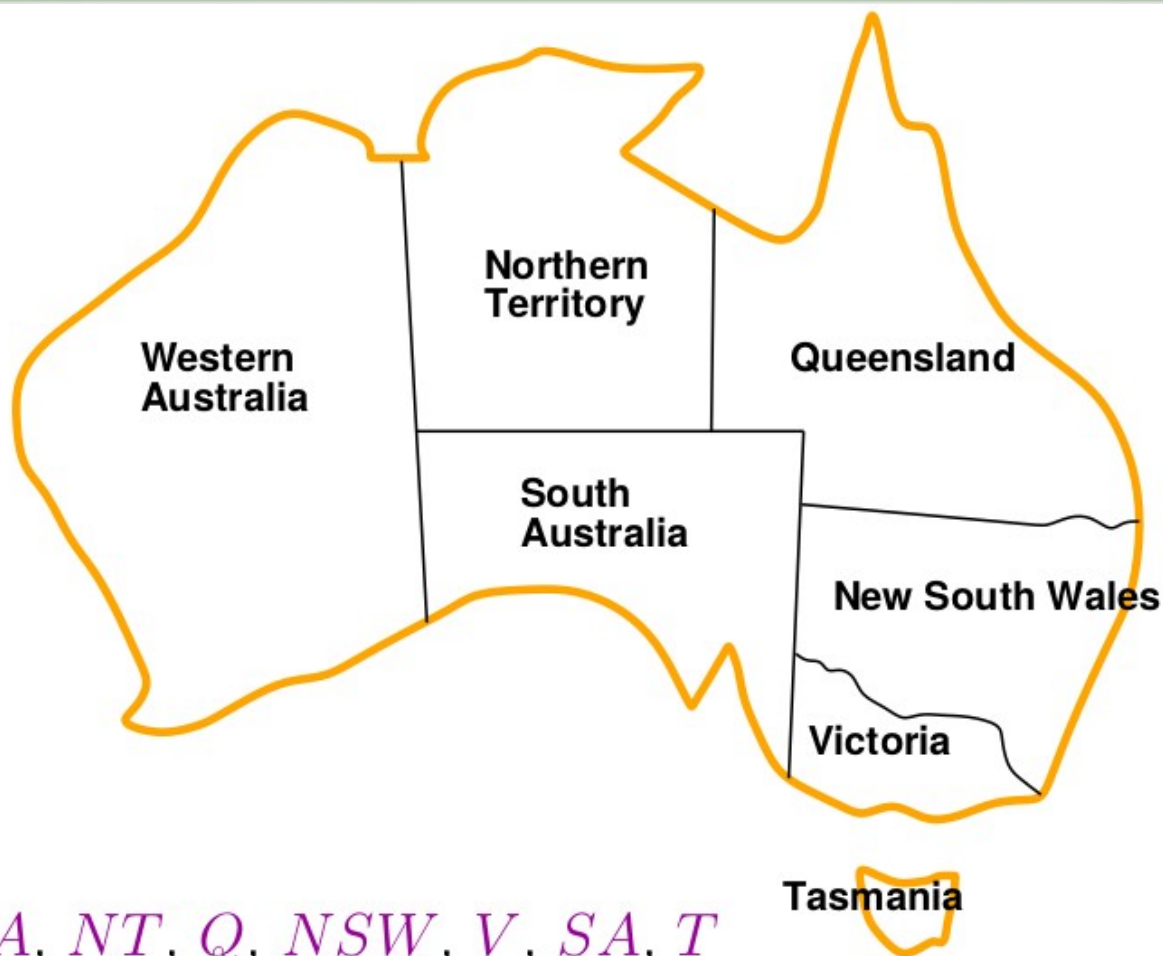
Problema de Satisfação de Restrição

- Resolução de PSR:
 - Definir espaço de estados;
 - Noção de solução;
 - Estado é definido por uma atribuição de valor a alguma ou todas as variáveis
 $\{ X_i = V_i ; X_j = V_j \};$
 - Em geral NP-Completo
 - Capaz de modelar diversos problemas na IA:
 - Alocação de recursos;
 - Coloração de grafos;
 - Agendamento de tarefas

Problema de Satisfação de Restrição

- **Atribuição Consistente:** não viola quaisquer restrições;
- **Atribuição Completa:** em que cada variável é atribuída um valor;
- **Solução em PSR:** é uma atribuição consistente e completa;
- **Atribuição parcial:** é aquela que atribui valores para apenas algumas das variáveis.

Coloração de Mapa



Variables WA, NT, Q, NSW, V, SA, T

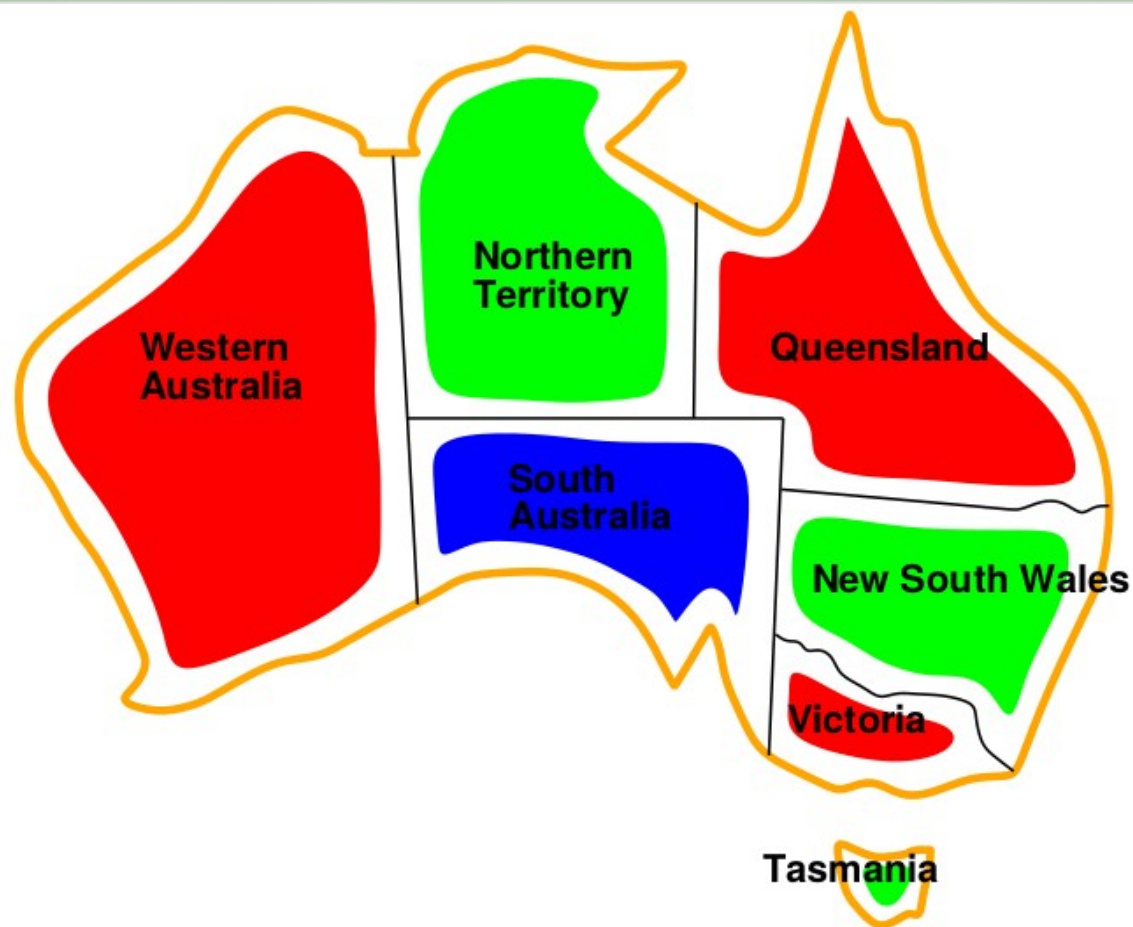
Domains $D_i = \{red, green, blue\}$

Constraints: adjacent regions must have different colors

e.g., $WA \neq NT$ (if the language allows this), or

$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \dots\}$

Coloração de Mapa



Soluções: são atribuições completas e consistentes.

$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

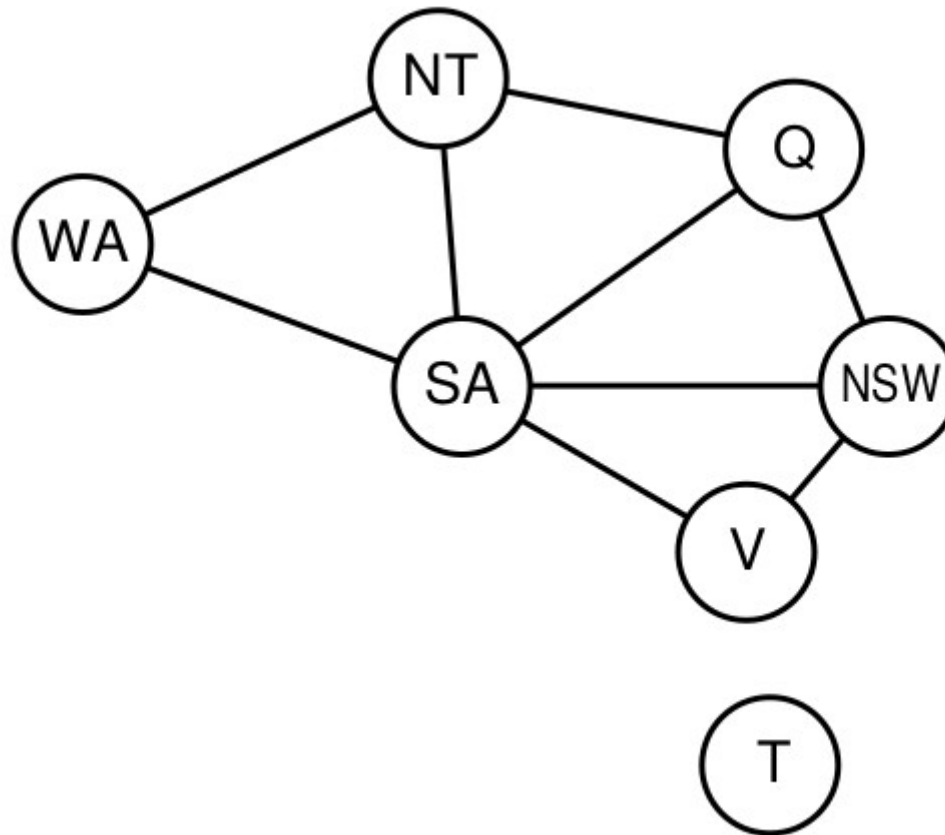
Problema de Satisfação de Restrição

- **Porque formular um problema como PSR?**

1. Natureza do próprio problema;
2. Se PSR, será mais fácil resolvê-lo;
3. Mais rápido que algoritmos de busca (limita espaço do problema).
 - Se restringir **AZUL**, $3^5=243 \rightarrow 2^5=32$ atribuições;
4. Difere das buscas, pois aborta caminhos não promissores;
5. Pode resolver problemas intratáveis para os algoritmos de busca em espaços.

Grafo de Restrição

- **PSR binário:** cada restrição relacionada com no máximo duas variáveis;
- **Grafo de Restrição:** nós são variáveis e aresta são as restrições.



Variáveis

- **Domínios Discretos e Finitos/Infinitos:**
 - 8-rainhas (cada rainha recebe valor entre 1 e 8)
 - Agendamento de tarefas ($\text{eixo_frente} + 5 \leq \text{Roda_frente}$)
- **Domínios Contínuos:**
 - Problemas de programação linear que as restrições são equações e inequações que formam uma região convexa;
 - Problemas resolvidos em tempo polinomial ao número de variáveis.

Tipos de Restrições

- **Unárias:** $\langle (SA), SA \neq \text{verde} \rangle$;
- **Binárias:** $SA \neq NSW$;
- **Global:** envolve número arbitrário de variáveis. Por exemplo:

SUDOKU

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

- **Absolutas:** a violação elimina uma solução potencial. Por Exemplo, 1 professor não pode lecionar em 2 salas ao mesmo tempo;
- **De Preferência:** indicam as soluções preferidas.. Por exemplo, professor prefere não lecionar pela manhã.

Busca em Profundidade com Retrocesso

- *Atribuição comutativa:*

$WA = \text{red} \text{ então } NT = \text{green} \leftrightarrow NT = \text{green} \text{ então } WA = \text{red}$

- Apenas precisa considerar atribuição para uma única variável/nó por vez:

$\Rightarrow b = d$ e existe d^n folhas

- Conhecida como **Backtracking Search**

Backtracking Search

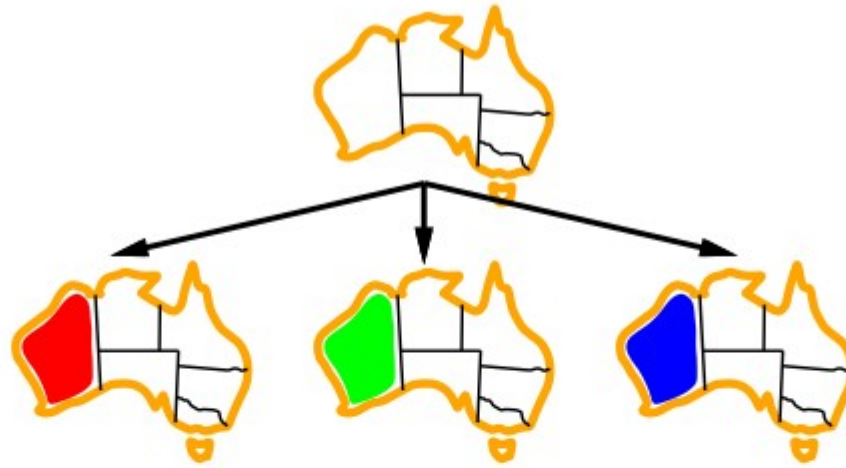
Função ProcuraRetrocesso (*csp*) **devolve** *solução* ou *falha*
devolve ProcuraRetrocessoRecursiva(*{}*, *csp*)

Função ProcuraRetrocessoRecursiva(*atrib*, *csp*)
devolve *solução* ou *falha*
 se *atrib* está completa **então devolve** *atrib*
 var ← SeleccionaVariávelNãoAtribuída(*Variáveis*[*csp*], *atrib*, *csp*)
 paracada *valor* em OrdenaValores(*var*, *atrib*, *csp*)
 se *valor* consistente com *atrib* dadas *Restrições*[*csp*] **então**
 adiciona {*var*=*valor*} a *atrib*
 resultado ← ProcuraRetrocessoRecursiva(*atrib*, *csp*)
 se *resultado* ≠ *falha* **então devolve** *resultado*
 remove {*var*=*valor*} de *atrib*
 devolve *falha*

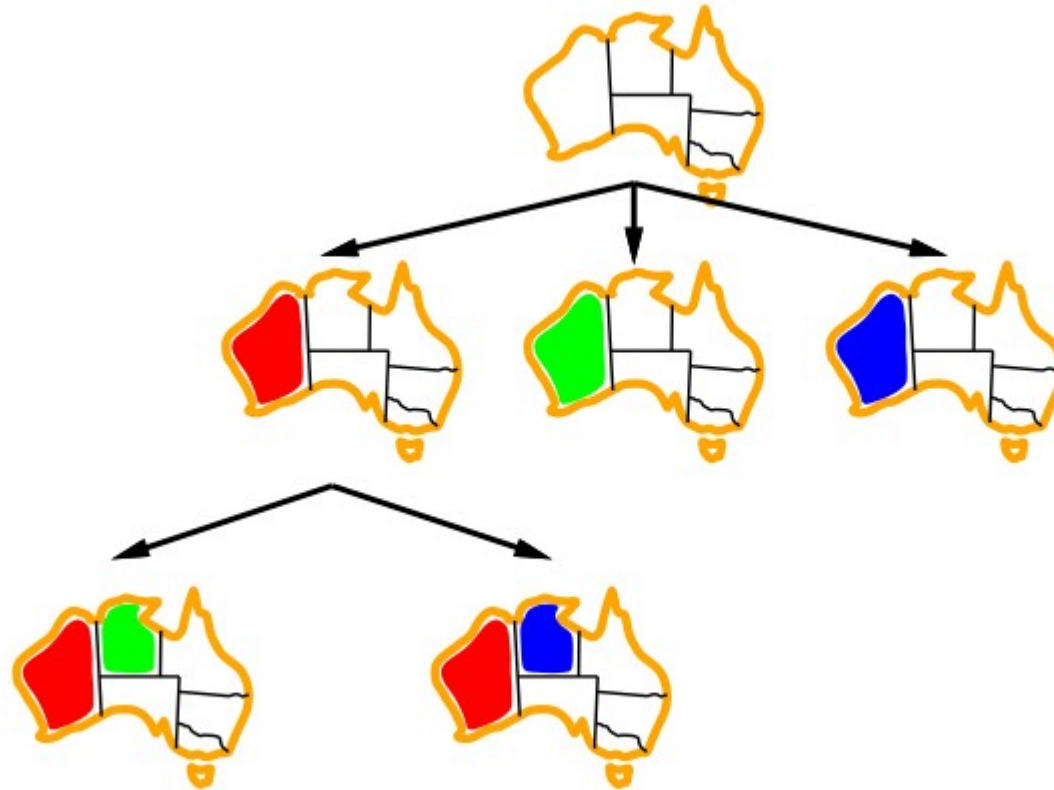
Backtracking Search



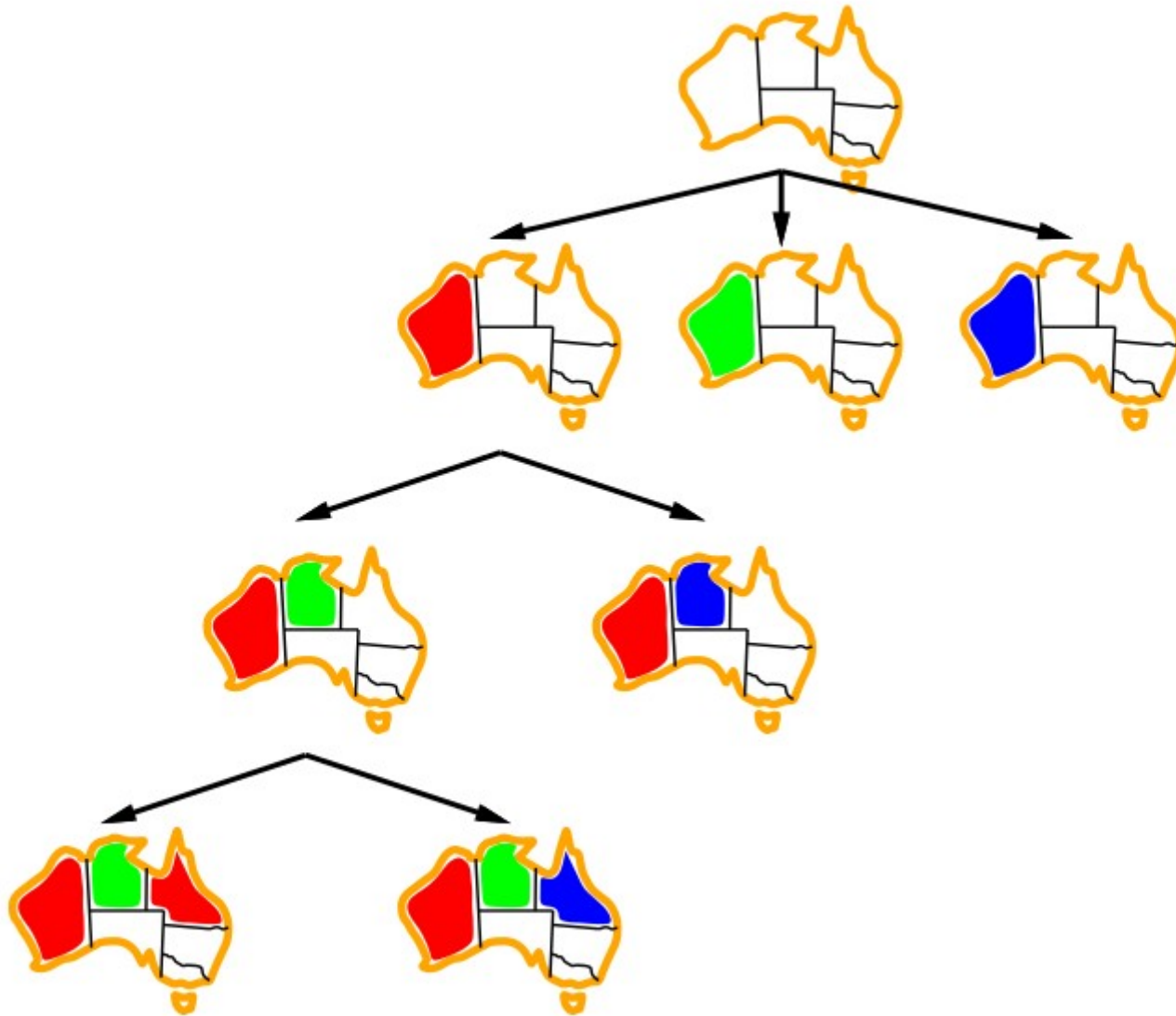
Backtracking Search



Backtracking Search



Backtracking Search



Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- **Heurística de Maior grau**
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

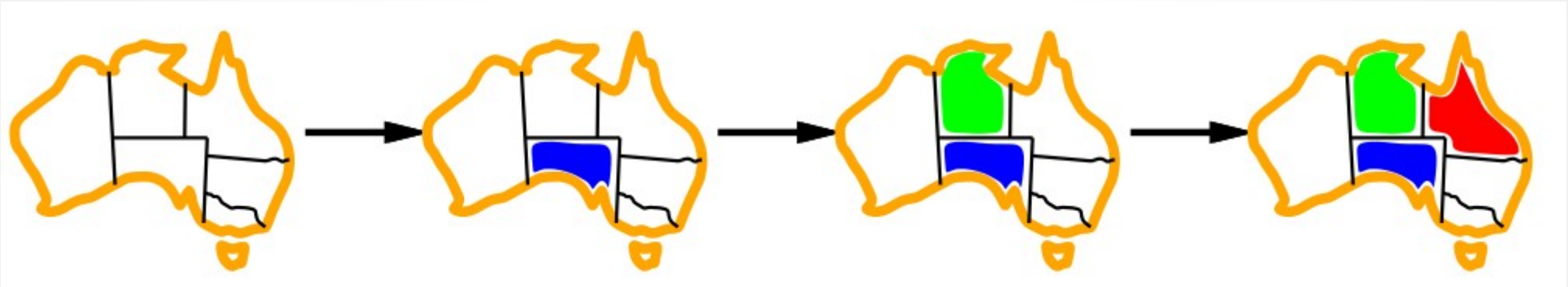
3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

Maior Grau

- *Tenta reduzir o fator de ramificação em escolhas futuras;*



Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- ***Heurística dos valores restantes mínimos***

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

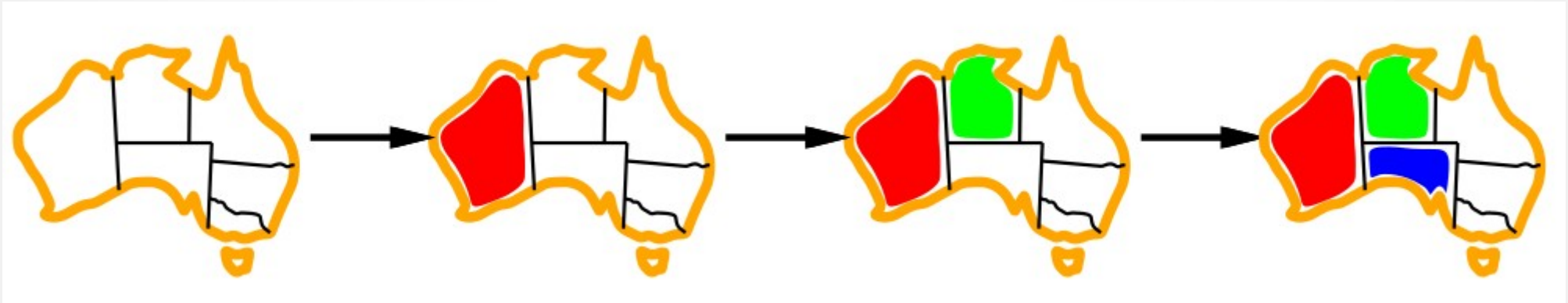
3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

Valores Restantes Mínimos

- *É escolher a variável com o menor número de valores legais;*



Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- ***Heurística do valor menos restritivo***

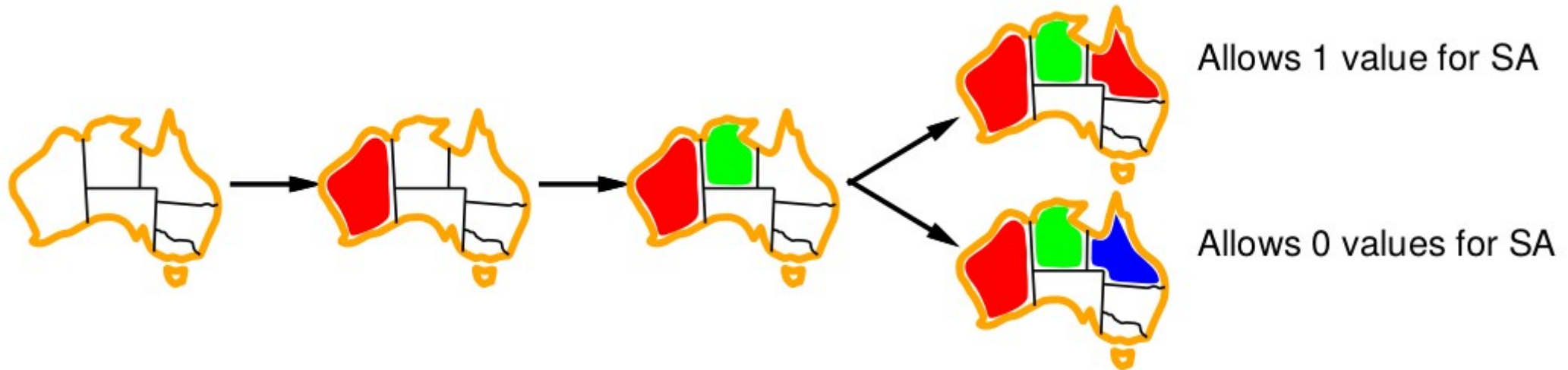
3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

Valor Menos Restritivo

- *É escolher o valor que menos prejudica as outras variáveis;*



Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

3. Nós podemos detectar falhas inevitáveis com antecedência?

- ***Verificação para frente***
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

Verificação para Frente



WA

NT

Q

NSW

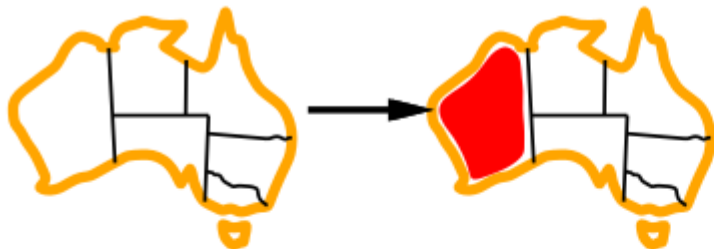
V

SA

T



Verificação para Frente



WA

NT

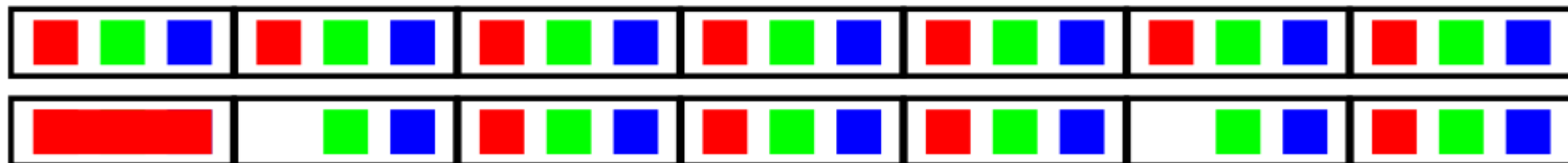
Q

NSW

V


SA

T




























































Verificação para Frente



WA	NT	Q	NSW	V	SA	T
						
						
						

Verificação para Frente



WA	NT	Q	NSW	V	SA	T
  	  	  	  	  	  	  
	 	  	  	  	 	  
			 	  		  
						  

Verificação para Frente



WA	NT	Q	NSW	V	SA	T
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>



Falhou!

Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

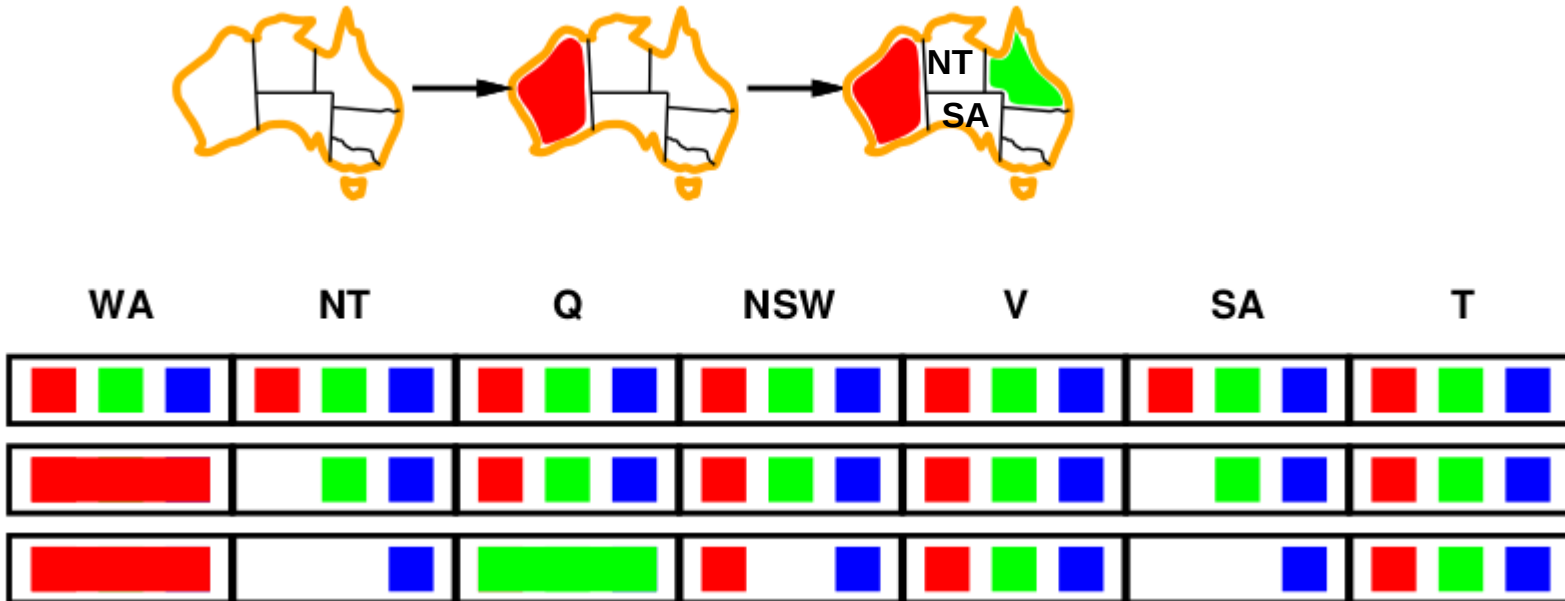
3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- **Propagação de restrições**

4. Nós podemos tirar vantagens da estrutura do problema?

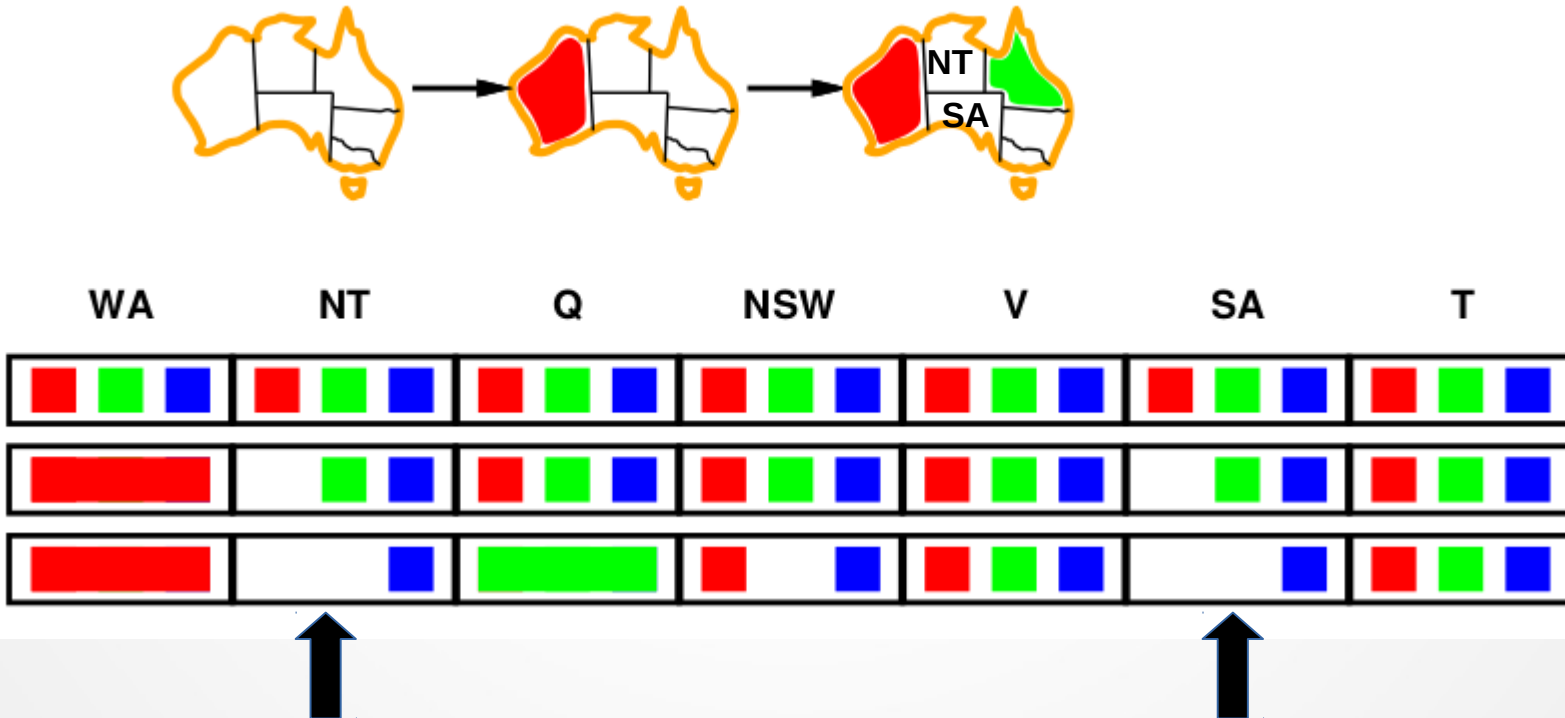
Propagação de restrições

- **Verificação para frente** propaga informação de variáveis já atribuídas para não-atribuídas;
- Entretanto, ele não fornece detecção de falhas com antecedência.



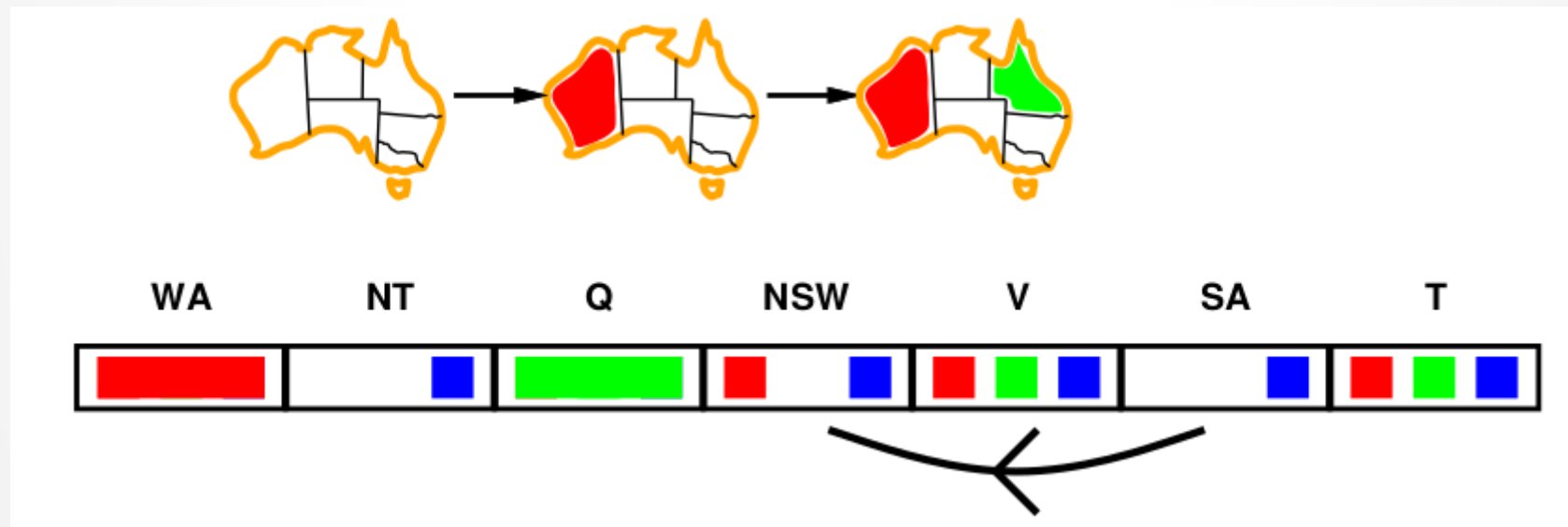
Propagação de restrições

- **Verificação para frente** propaga informação de variáveis já atribuídas para não-atribuídas;
- Entretanto, ele não fornece detecção de falhas com antecedência.



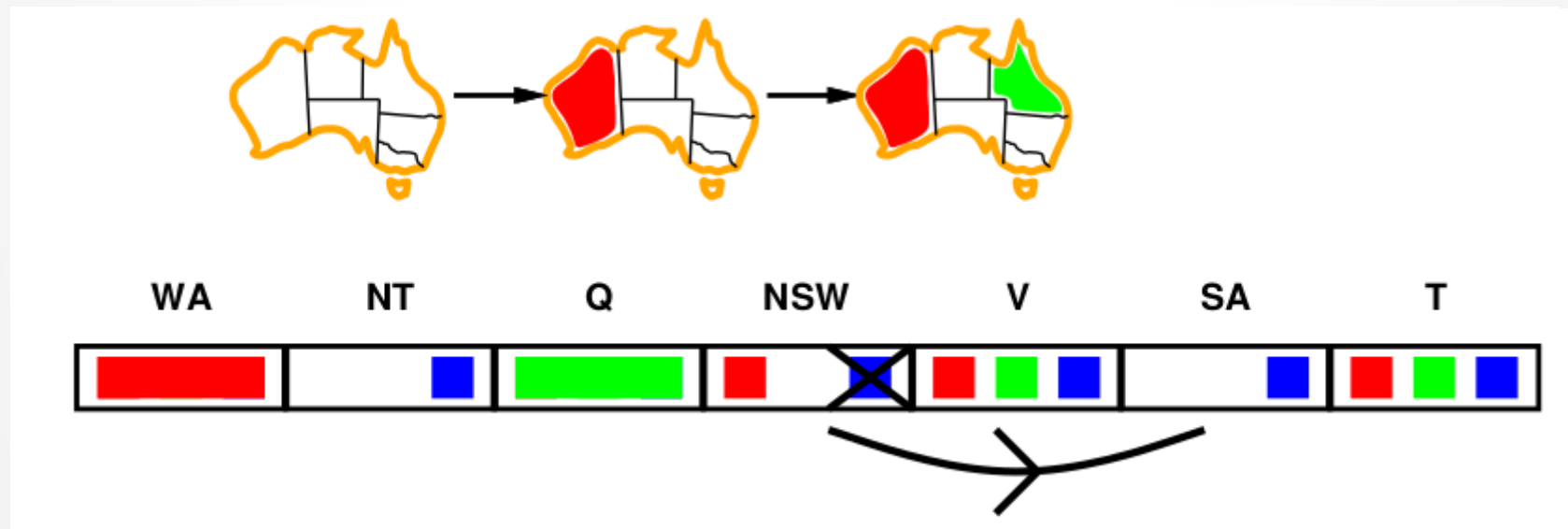
Propagação de restrições

- **Consistência de Arco:** Mais simples forma de propagar criando arco consistente, i.e., para todo valor possível, existe algum que poderá ser legalmente atribuído (respeita restrição).



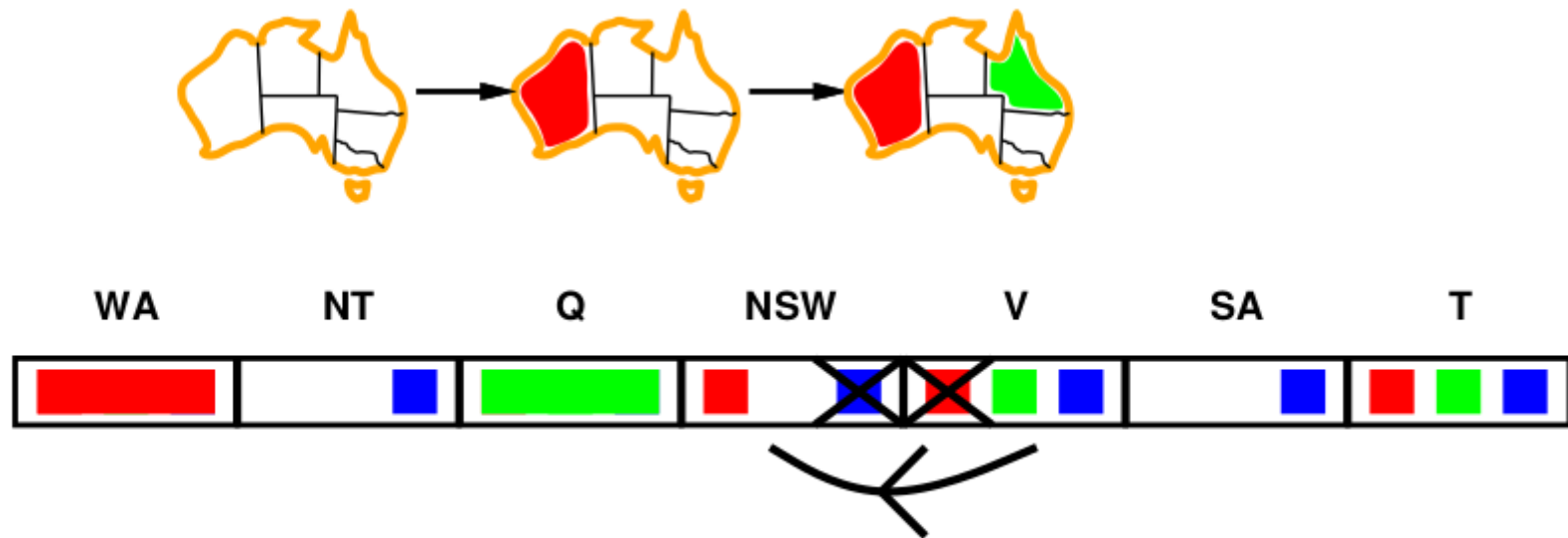
Propagação de restrições

- **Consistência de Arco:** Mais simples forma de propagar criando arco consistente, i.e., para todo valor possível, existe algum que poderá ser legalmente atribuído (respeita restrição).



Propagação de restrições

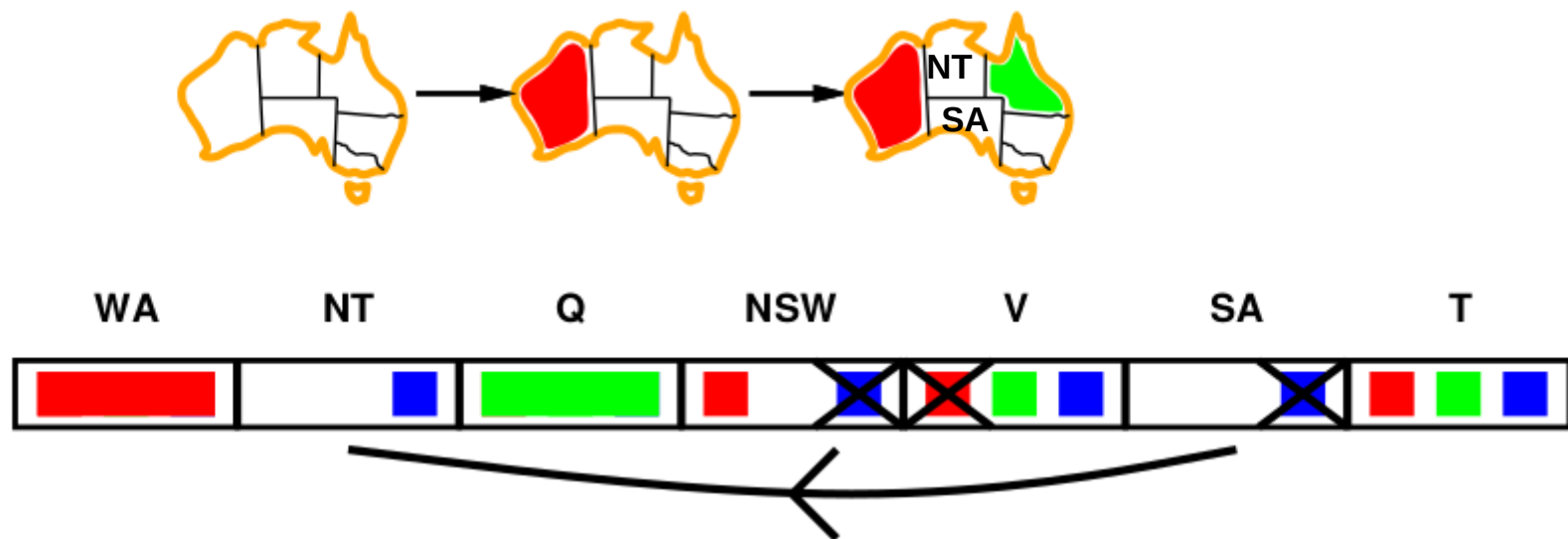
- **Consistência de Arco:** Mais simples forma de propagar criando arco consistente, i.e., para todo valor possível, existe algum que poderá ser legalmente atribuído (respeita restrição).



If X loses a value, neighbors of X need to be rechecked

Propagação de restrições

- **Consistência de Arco:** Mais simples forma de propagar criando arco consistente, i.e., para todo valor possível, existe algum que poderá ser legalmente atribuído (respeita restrição).



If X loses a value, neighbors of X need to be rechecked

Arc consistency detects failure earlier than forward checking

Melhoramentos do Backtracking Search

1. Qual variável deve ser atribuída a seguir?

- *Heurística de Maior grau*
- *Heurística dos valores restantes mínimos*

2. Em qual ordem os valores devem ser atribuídos?

- *Heurística do valor menos restritivo*

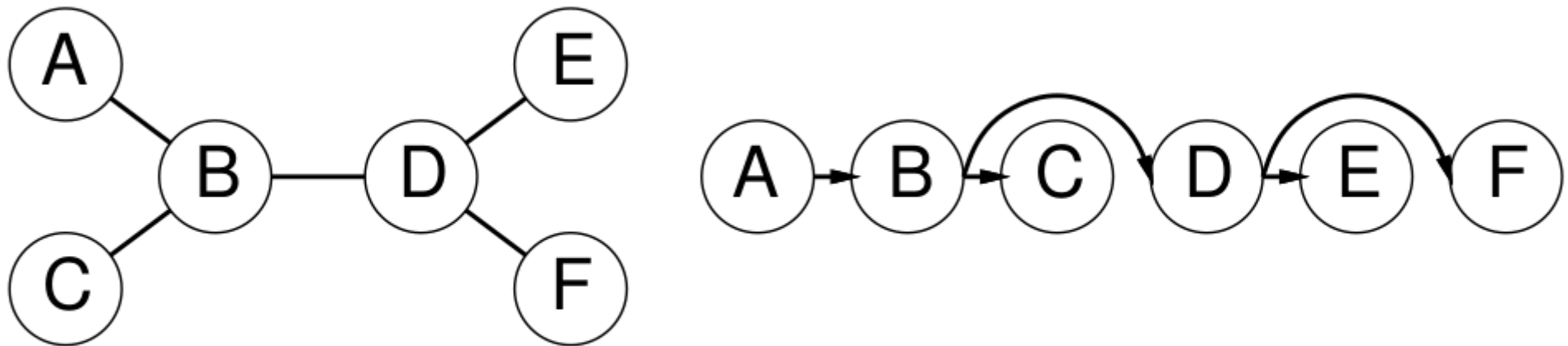
3. Nós podemos detectar falhas inevitáveis com antecedência?

- *Verificação para frente*
- *Propagação de restrições*

4. Nós podemos tirar vantagens da estrutura do problema?

PSR estruturado em árvore

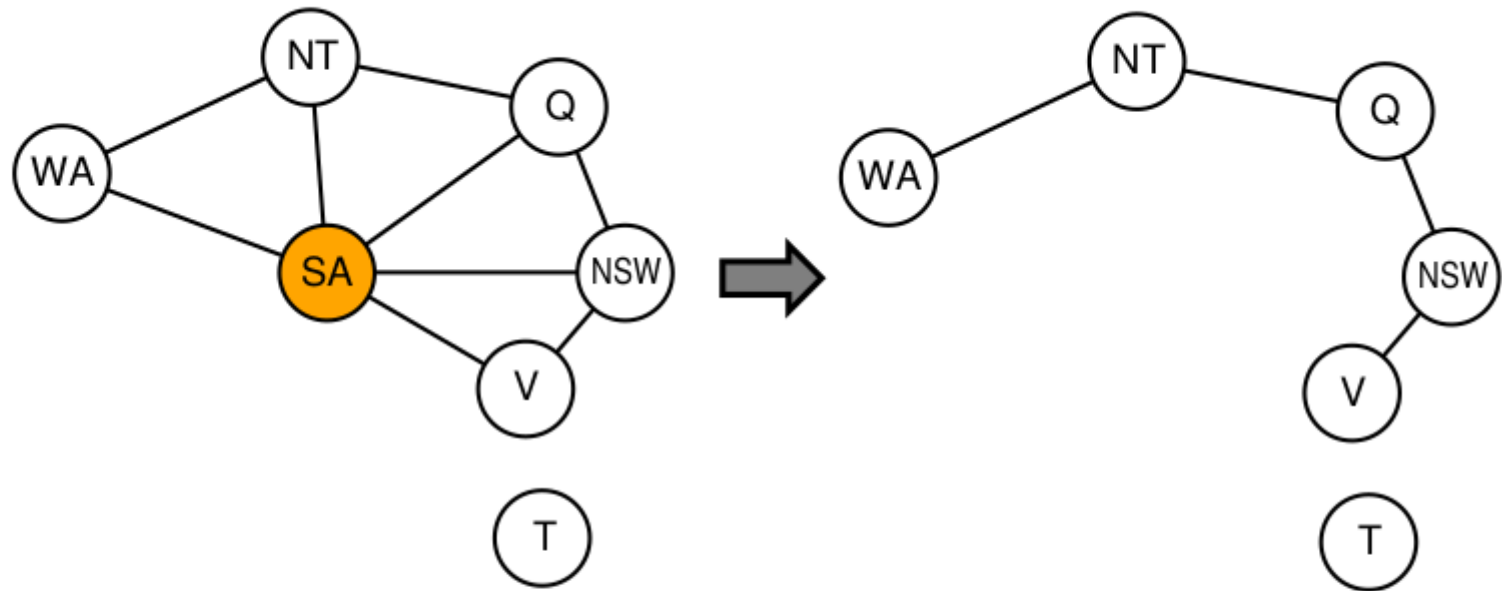
1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering



2. For j from n down to 2, apply REMOVEINCONSISTENT($Parent(X_j), X_j$)
3. For j from 1 to n , assign X_j consistently with $Parent(X_j)$

PSR estruturado em árvore

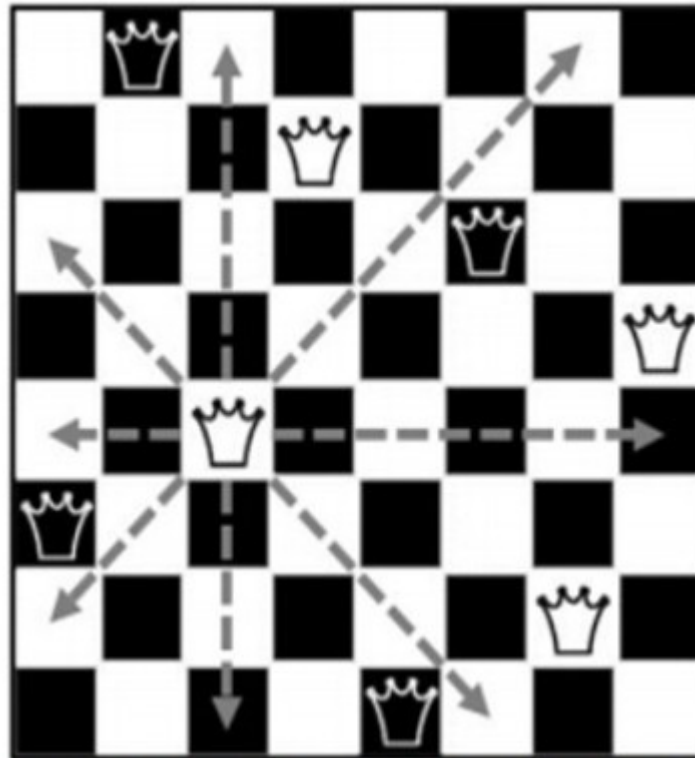
Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

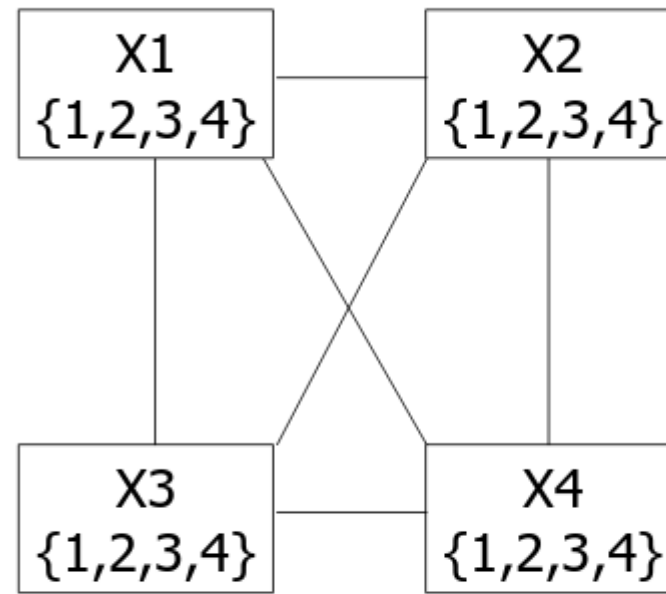
Cutset size $c \Rightarrow$ runtime $O(d^c \cdot (n - c)d^2)$, very fast for small c

Como modelar n-rainhas como PSR?



Backtracking + Verificação para Frente

	1	2	3	4
1				
2				
3				
4				

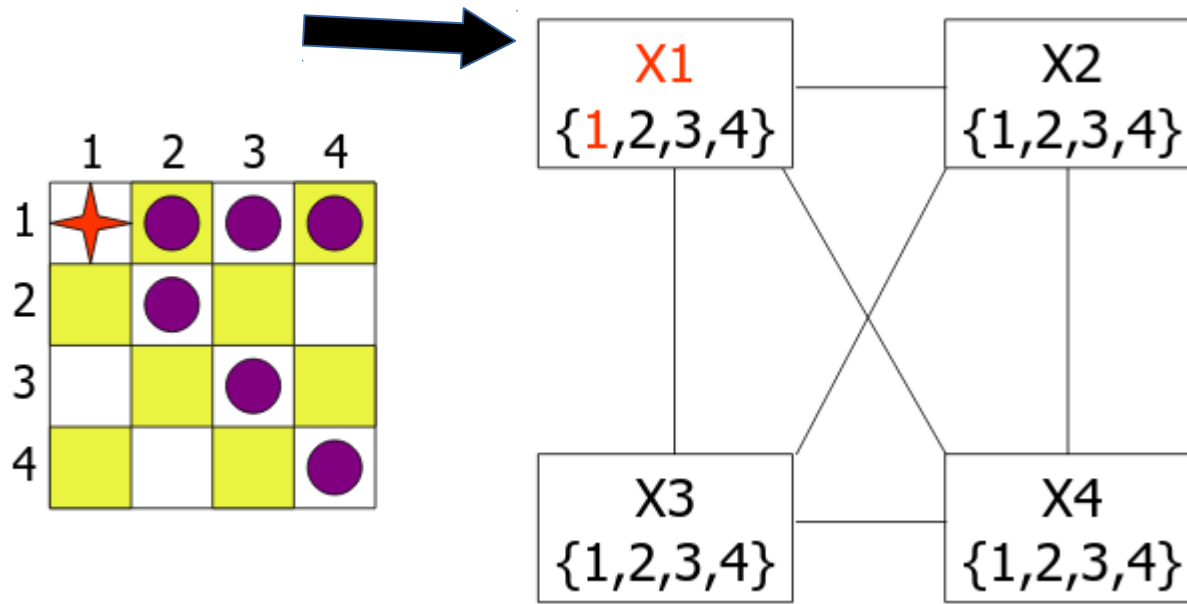


Variáveis $\{X1, X2, X3, X4\}$

Domínio $X1 = \text{Dom } X2 = \text{Dom } X3 = \text{Dom } X4 = \{1,2,3,4\}$

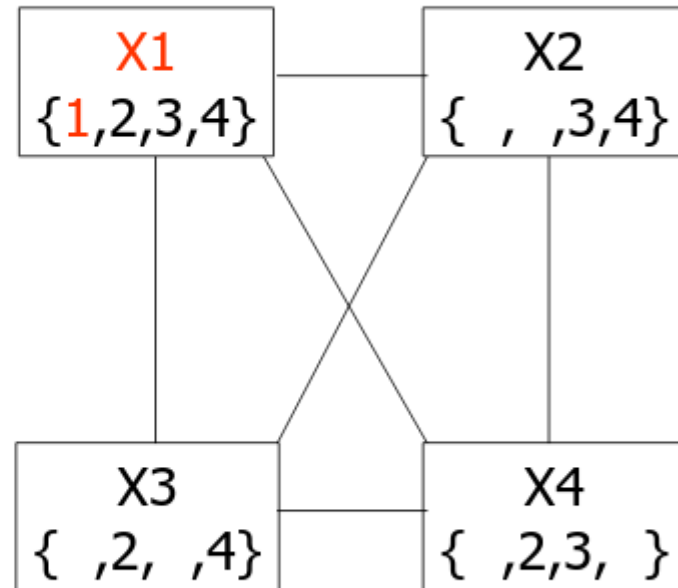
$X_j = i$ significa que rainha j está na posição (i,j)

Backtracking + Verificação para Frente



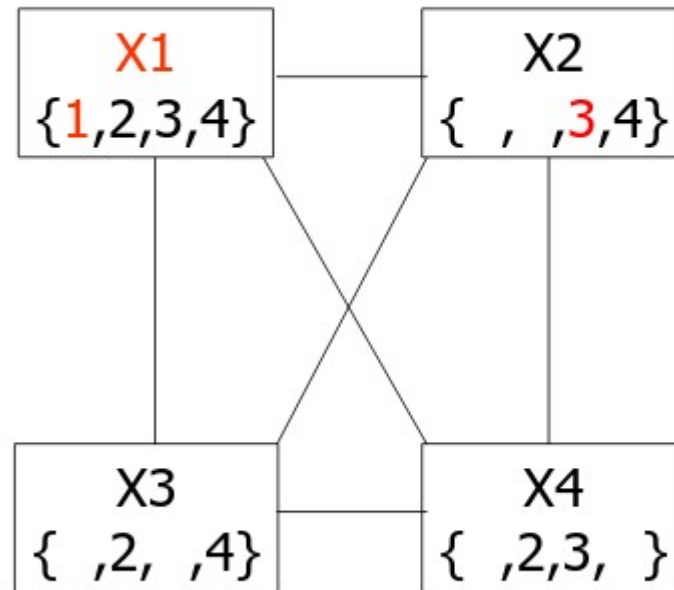
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●		
3			●	
4				●



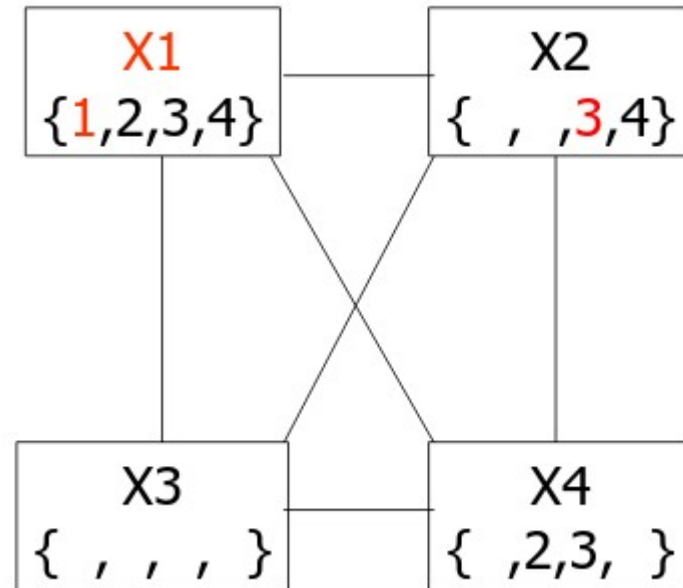
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●	●	
3		★	●	●
4			●	●



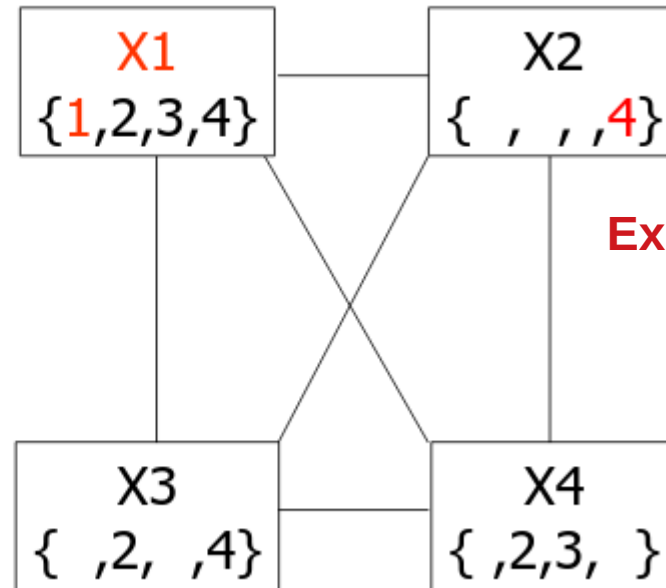
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●	●	
3		★	●	●
4			●	●



Backtracking + Verificação para Frente

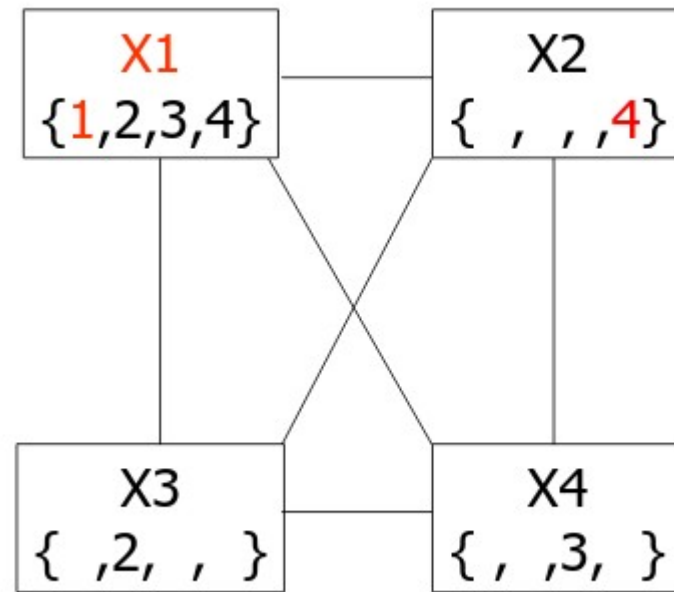
	1	2	3	4
1	★	●	●	●
2		●		●
3			●	
4		★	●	●



Existe outro valor?

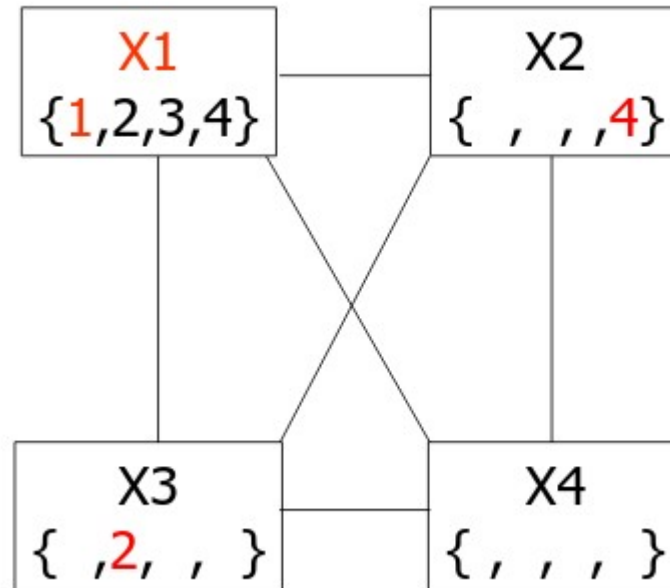
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●		●
3			●	
4		★	●	●



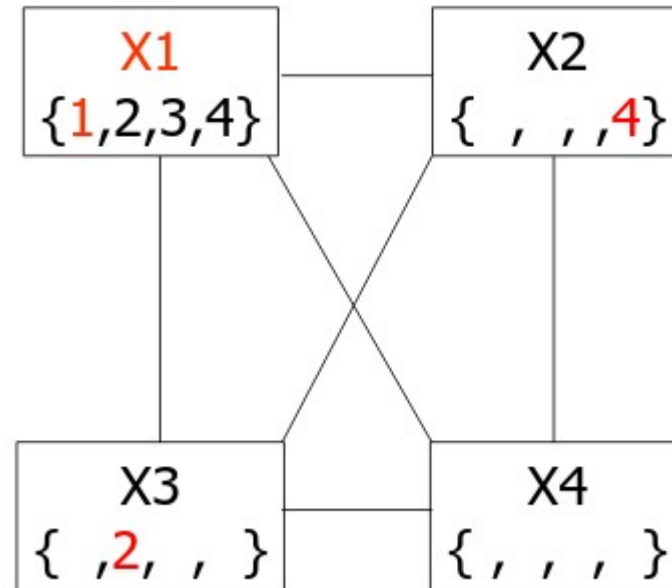
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	●
4		★	●	●



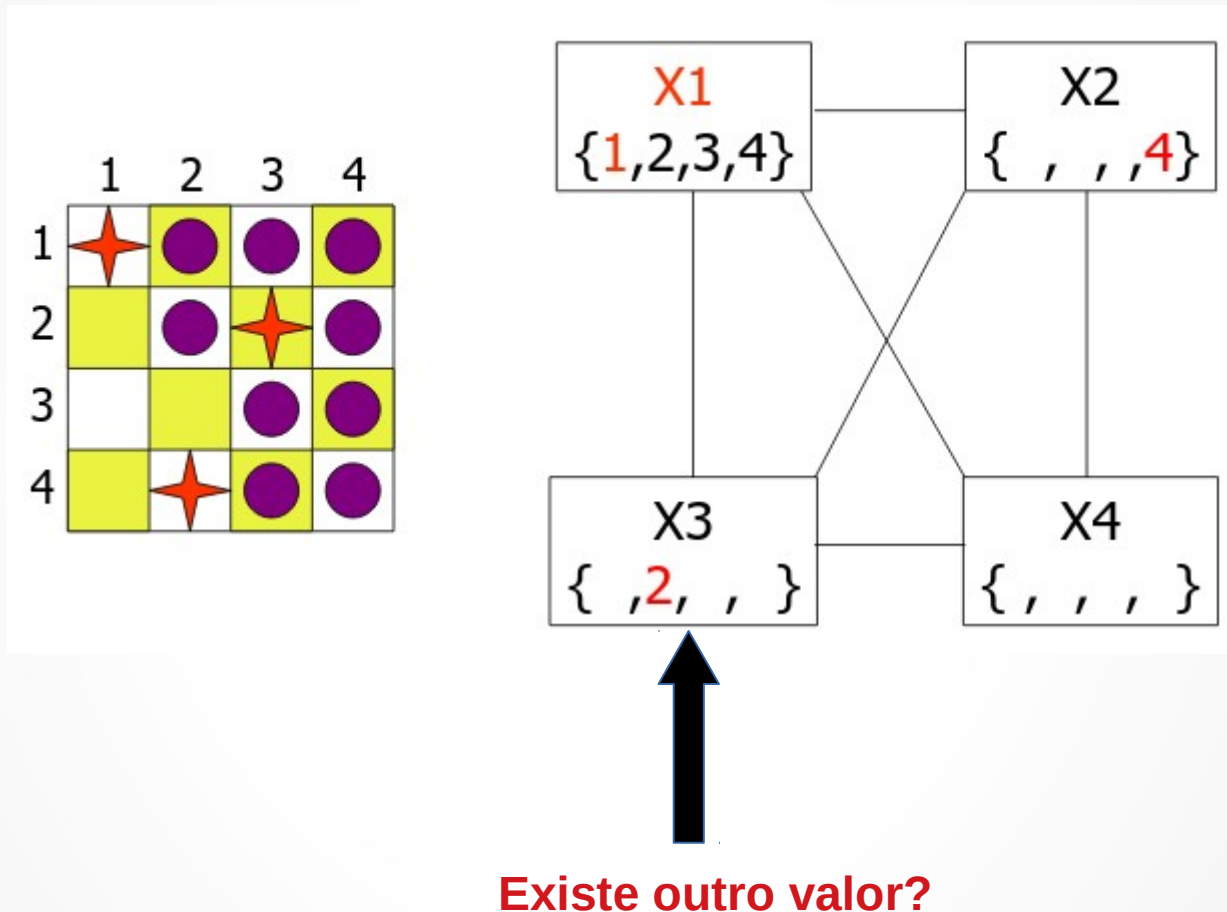
Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	●
4		★	●	●



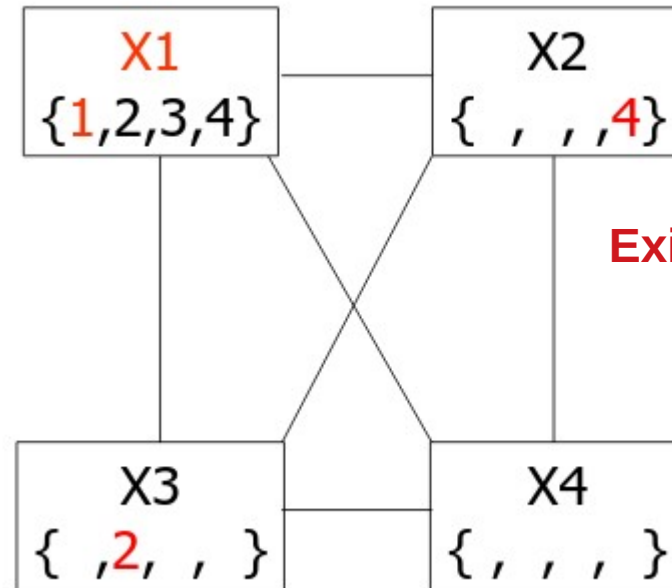
Falhou!

Backtracking + Verificação para Frente



Backtracking + Verificação para Frente

	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	●
4		★	●	●

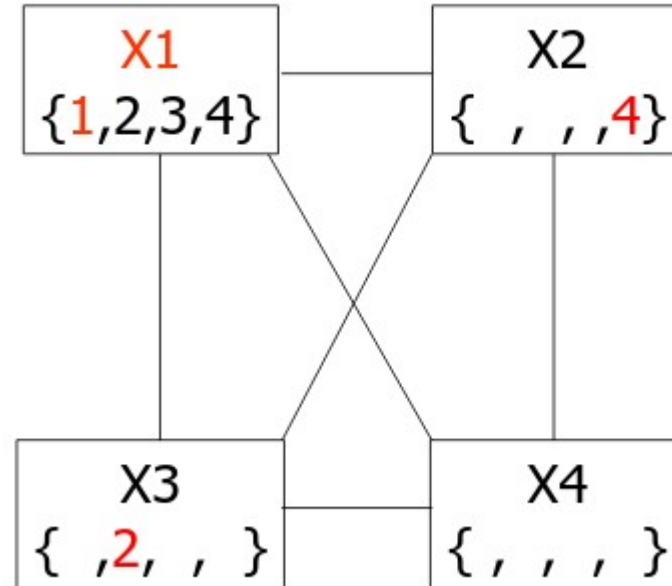


Backtracking + Verificação para Frente

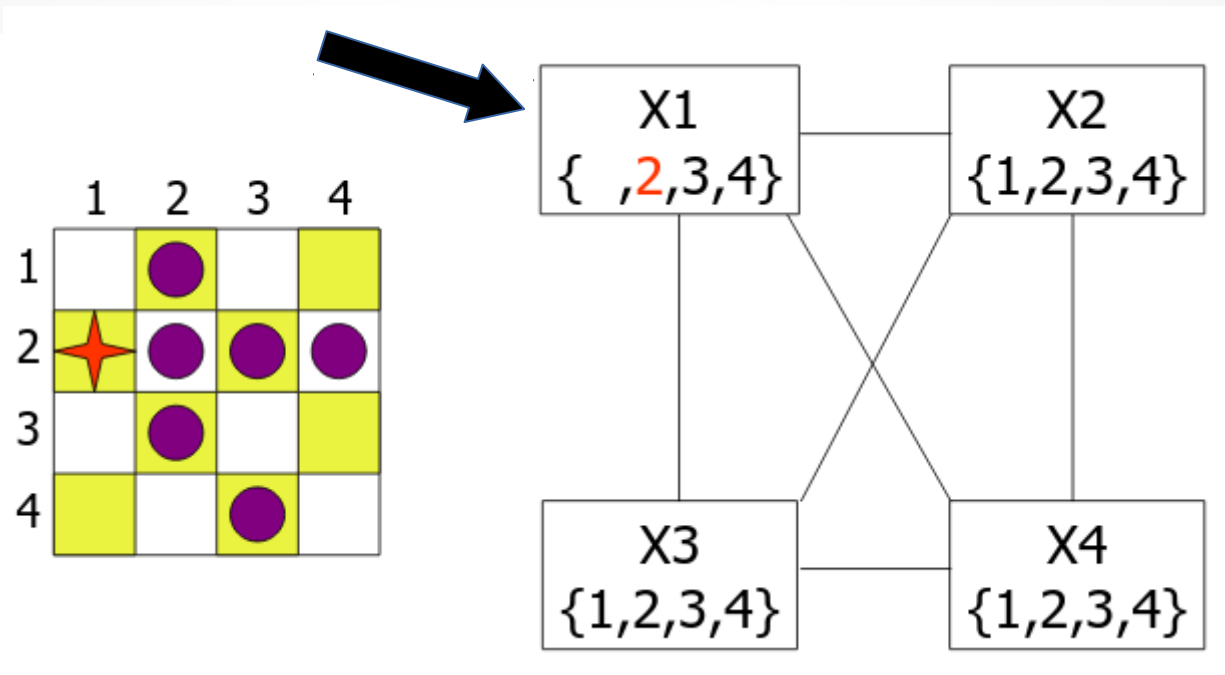
Existe outro valor?



	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	●
4		★	●	●

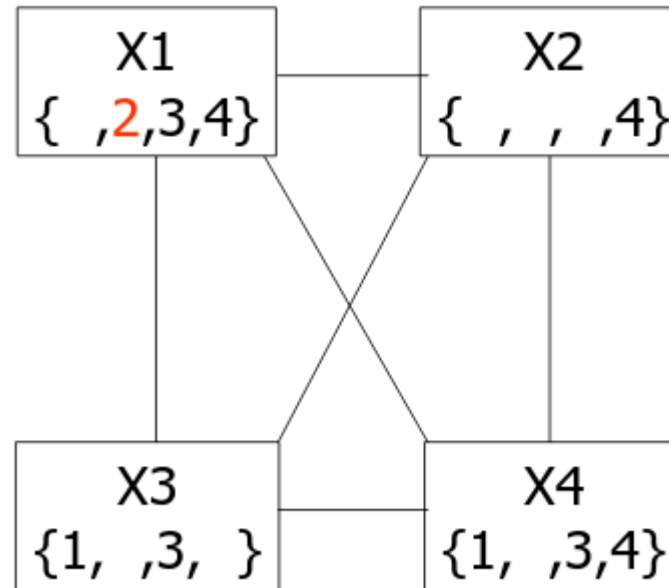


Backtracking + Verificação para Frente



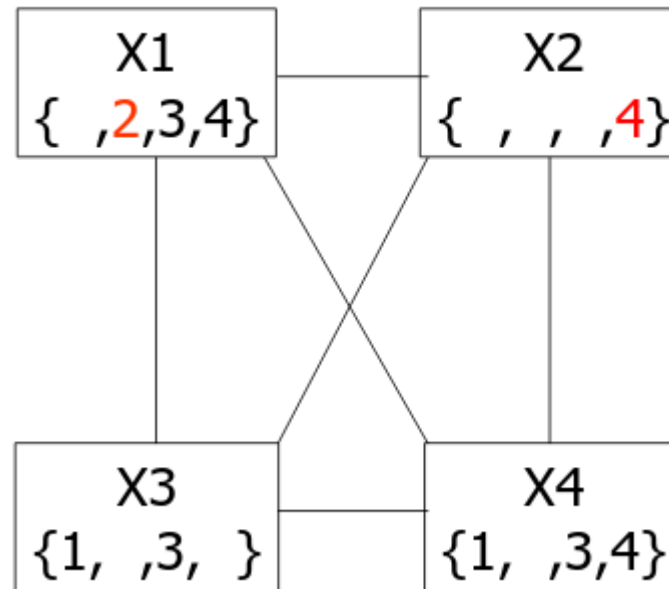
Backtracking + Verificação para Frente

	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



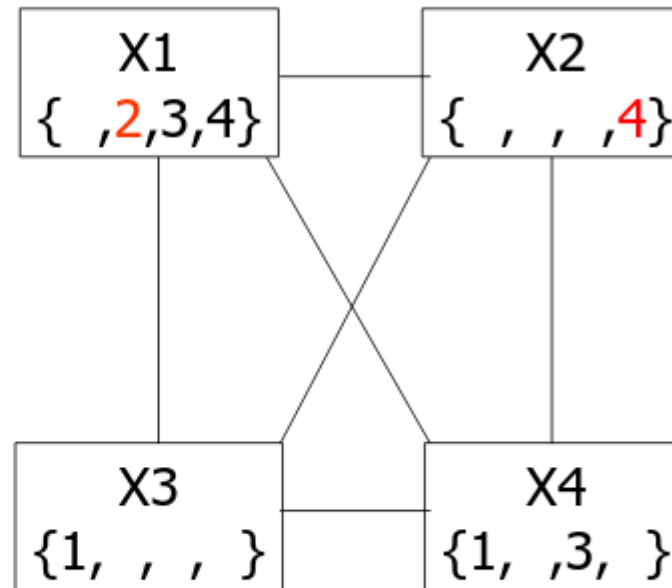
Backtracking + Verificação para Frente

	1	2	3	4
1		●		
2	★	●	●	●
3		●	●	
4		★	●	●



Backtracking + Verificação para Frente

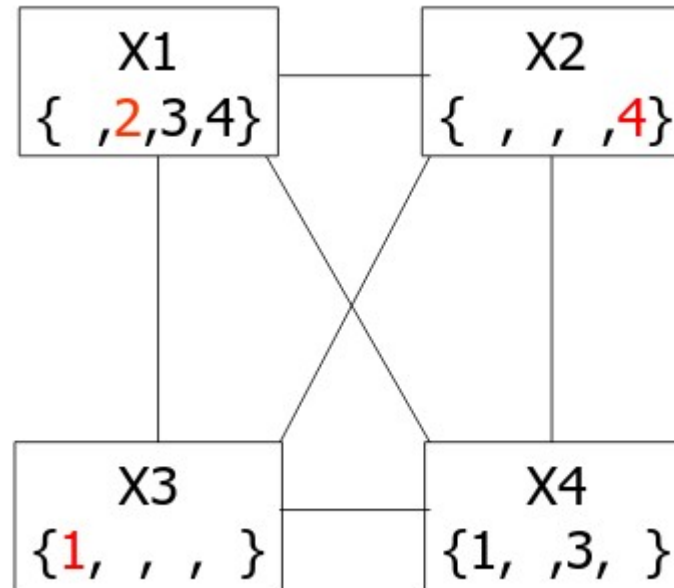
	1	2	3	4
1		●		
2	★	●	●	●
3		●	●	
4		★	●	●



Consistência de arcos já teria encontrado a solução!

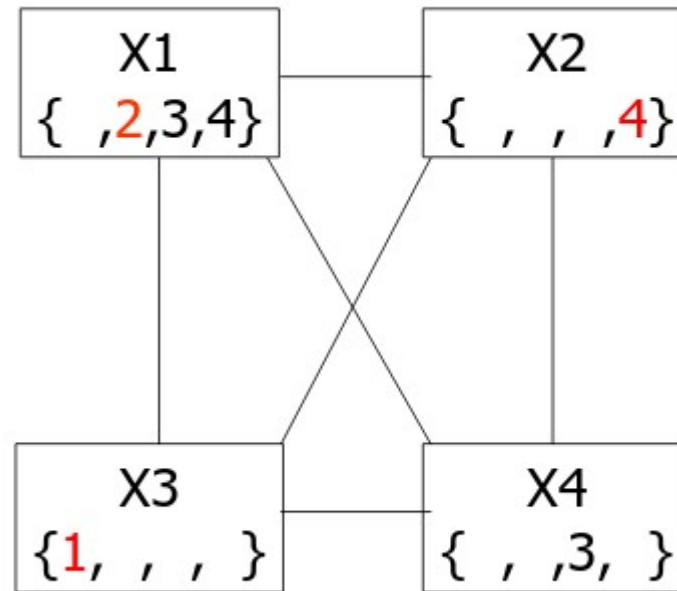
Backtracking + Verificação para Frente

	1	2	3	4
1		●	★	●
2	★	●	●	●
3		●	●	
4		★	●	●



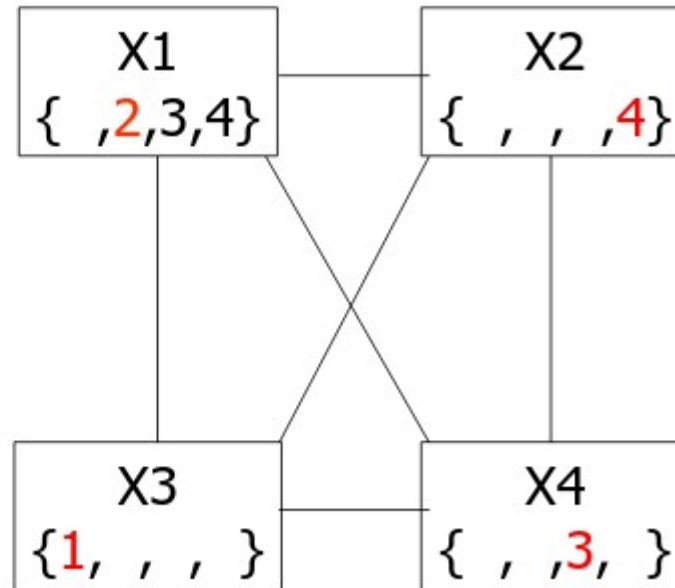
Backtracking + Verificação para Frente

	1	2	3	4
1		●	✱	●
2	✱	●	●	●
3		●	●	
4		✱	●	●



Backtracking + Verificação para Frente

	1	2	3	4
1		●	★	●
2	★	●	●	●
3		●	●	★
4		★	●	●



Solução encontrada!

Referências

- Cap 6 Livro Russel e Norvig
- Material online
<https://docplayer.com.br/12687100-Satisfacao-de-restricoes-capitulo-5-disponivel-online.html>UFPE
- Material complementar:
 - https://www.youtube.com/watch?v=_e64FiDWvqs
 - https://www.youtube.com/watch?v=Hv_JlWld9iQ