

CC7711

Inteligência Artificial e Robótica

Prof. Dr. Flavio Tonidandel

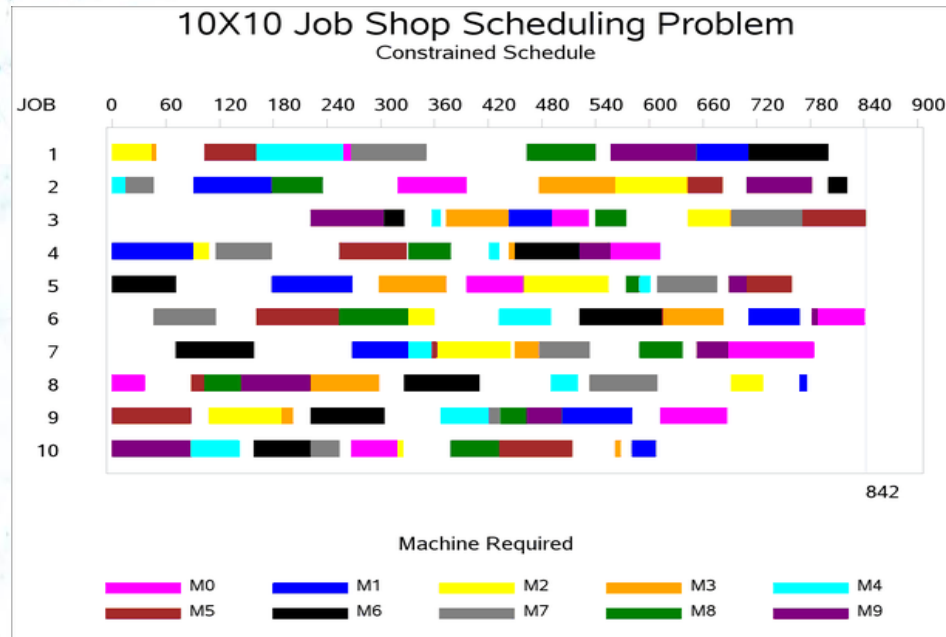
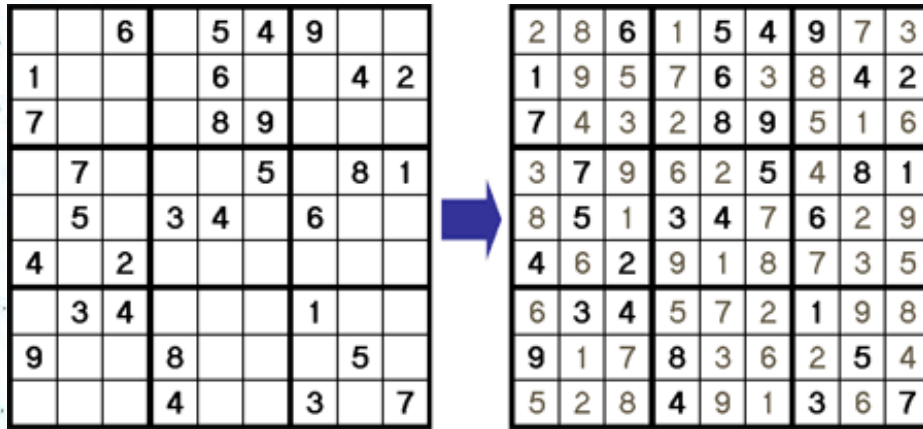


The background of the slide features a complex network diagram. It consists of numerous circular nodes of varying sizes, connected by thin, light-blue lines. The nodes are distributed across the entire frame, with a higher density on the left side, creating a sense of a vast, interconnected system. The overall color palette is light blue and white, giving it a clean, technical appearance.

Problemas de Satisfação de Restrições

Constraint Satisfaction Problems (CSP)

Sudoku



Employee shift rostering

Populate each work shift with a nurse.

Maternity nurses			Emergency nurses			Basic nurses									
A	Ann	B	Beth	C	Cory	D	Dan	E	Elin	G	Greg	H	Hue	I	Ilse

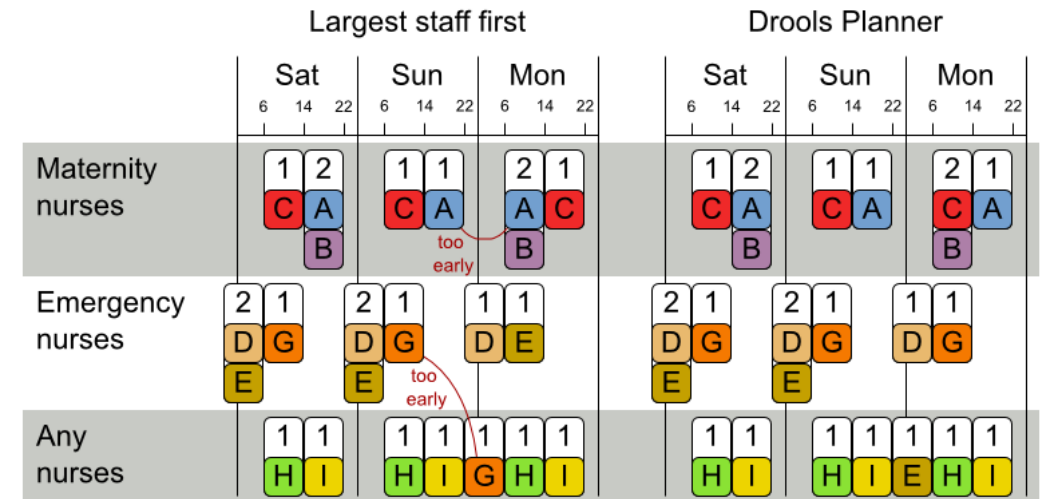


Image by Geoffrey De Smet - Java Zone

Image by support.sas.com

Constraint Satisfaction Problems - CSP

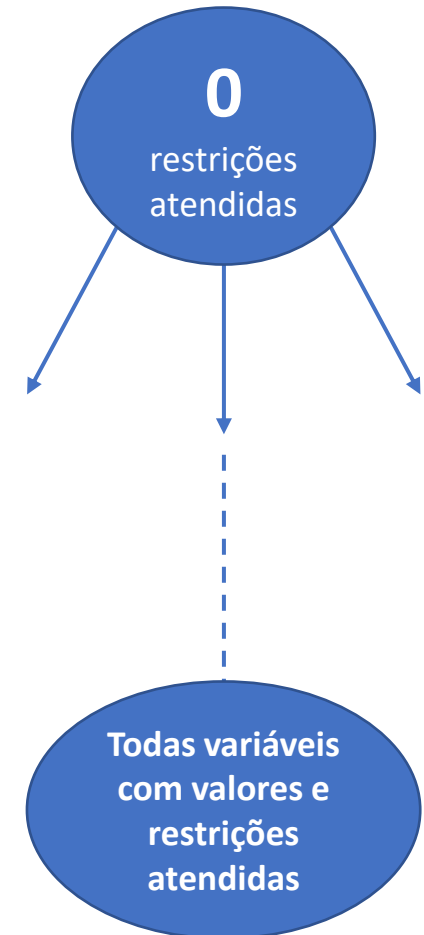
- Em português:
 - Problema de Satisfação de Restrições – PSR
- Problemas CSP:
 - Obedecem a uma representação específica
 - É possível determinar heurísticas que tiram proveito desta representação:
 - para resolver problemas CSP de forma mais eficiente que um sistema de busca

Constraint Satisfaction Problems - CSP

- Em termos formais, um CSP é definido por:
 - Variáveis: $\langle X_1, X_2, \dots, X_n \rangle$
 - Restrições: $\langle C_1, C_2, \dots, C_m \rangle$
 - Cada variável possui um Domínio D_i de valores possíveis
 - As restrições especificam as combinações permitidas
- O que é uma solução em CSP ?
 - Ter valores permitidos para TODAS as variáveis
- Podemos resolver um problema CSP com um sistema geral de busca ?
 - Sim. Mas como um problema CSP possui informações adicionais como restrições, podemos melhorar os sistemas de busca para esses problemas.

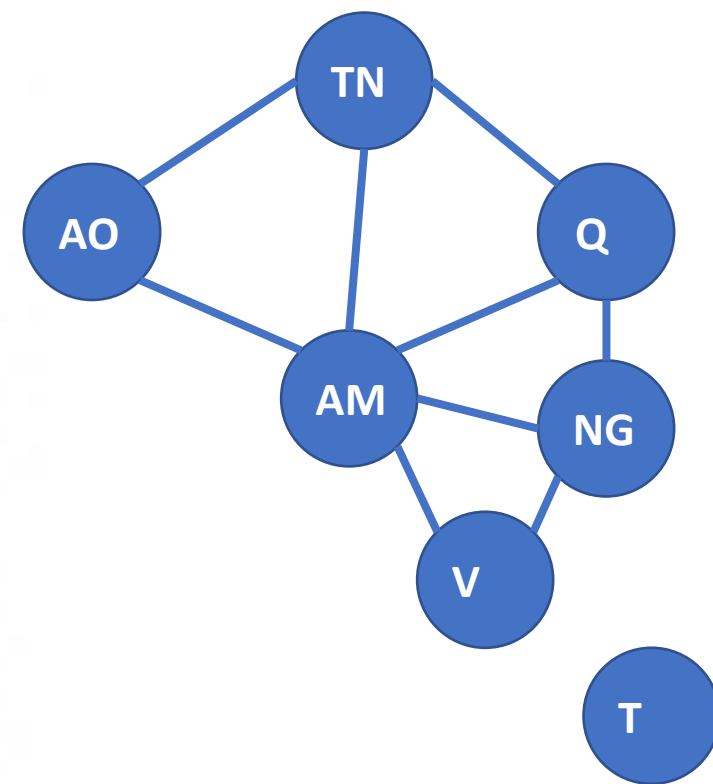
Principal característica de um CSP

- Não sabemos qual é o estado final !!!
- Queremos achar um estado que satisfaça as restrições impostas, QUALQUER UM !!
- PROCESSO DE BUSCA
 - É mais simples
 - Cada nível na árvore é uma valor atribuído à uma variável
 - Altura da árvore = num variáveis

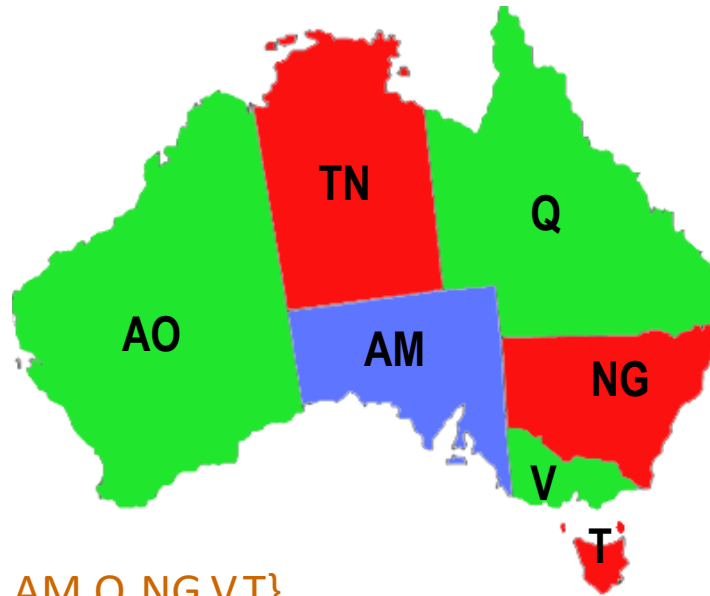


Exemplo: Coloração de Mapas

Mapa da Austrália



Exemplo: Coloração de Mapas



- 7 variáveis {AO,TN,AM,Q,NG,V,T}
- Cada variável tem o mesmo domínio {red, green, blue}
- Restrições: Nenhum nó adjacente pode ter a mesma cor:
 $AO \neq TN, AO \neq AM, TN \neq AM, TN \neq Q, AM \neq Q, AM \neq NG, AM \neq V, Q \neq NG, NG \neq V$

Variedades de Restrições

- Restrições unárias: envolve uma variável.
 - Ex: $AM \neq green$ ou $AM = blue$
- Restrições Binárias: envolve duas variáveis.
 - Ex: $AM \neq AO$
- Restrições de ordem maior: envolve 3 ou mais variáveis
- Pode-se ainda definir **Preferências**
 - São restrições fracas e servem para otimizar solução
 - e.g. *red* é melhor que *green*


Comutatividade do CSP

A ordem com que são atribuídos os valores das variáveis é **IRRELEVANTE**. Assim:

- Podemos gerar estados sucessores de um nó atribuindo um único valor a uma variável.
- Desde que esse valor não entre em conflito com outra variável já atribuída.

Não precisamos armazenar o caminho do estado inicial até o objetivo

- Estado inicial = vazio
- Estado Final = todas as variáveis atribuídas (caminho é IRRELEVANTE)



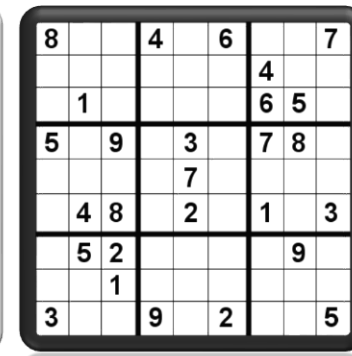
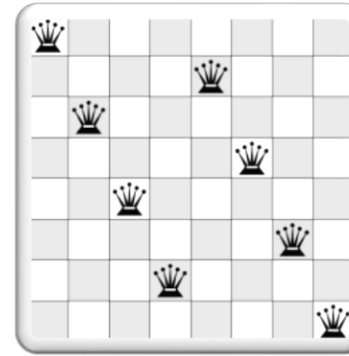
		6		5	4	9		
1				6			4	2
7				8	9			
	7				5		8	1
	5		3	4		6		
4		2						
	3	4				1		
9			8				5	
			4			3		7

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

Exemplos de CSP

Com solução polinomial

- 8-rainhas
- Sudoku
- Coloração de mapas
- Teste QI/Einstein



	1° Amiga	2° Amiga	3° Amiga	TESTE DE QI Teste do Einstein	
Nome	<input type="text" value="Bruna"/>	<input type="text" value="Marcela"/>	<input type="text" value="Carol"/>	<input type="text" value="Daniel"/>	<input type="text" value="Henrique"/>
Namorado	<input type="text" value="Luís"/>	<input type="text" value="Yuri"/>	<input type="text" value="Alex"/>	<input type="text" value="21"/>	<input type="text" value="24"/>
Idade	<input type="text" value="25"/>	<input type="text" value="22"/>	<input type="text" value="23"/>	<input type="text" value="Verde"/>	<input type="text" value="Azul"/>
Cor	<input type="text" value="Branco"/>	<input type="text" value="Amarelo"/>	<input type="text" value="Vermelho"/>	<input type="text" value="Comédia"/>	<input type="text" value="Ficção"/>
Filme	<input type="text" value="Drama"/>	<input type="text" value="Ação"/>	<input type="text" value="Romance"/>		

- NP-Completo
 - Problemas de Satisfabilidade
 - CSP booleanos
 - (encontrar valores V e F para tornar uma formula lógica verdadeira)

Vamos resolver um problema CSP

Vimos que os sistemas de busca podem melhorar sua eficiência com uso de heurísticas

- Heurística = conhecimento da aplicação

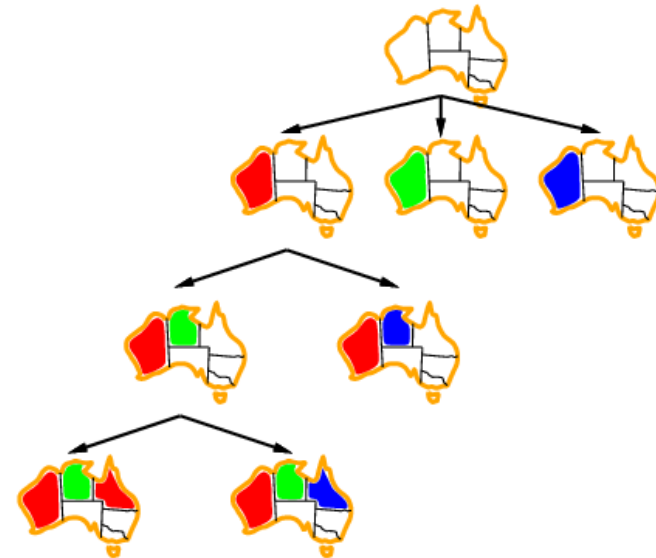
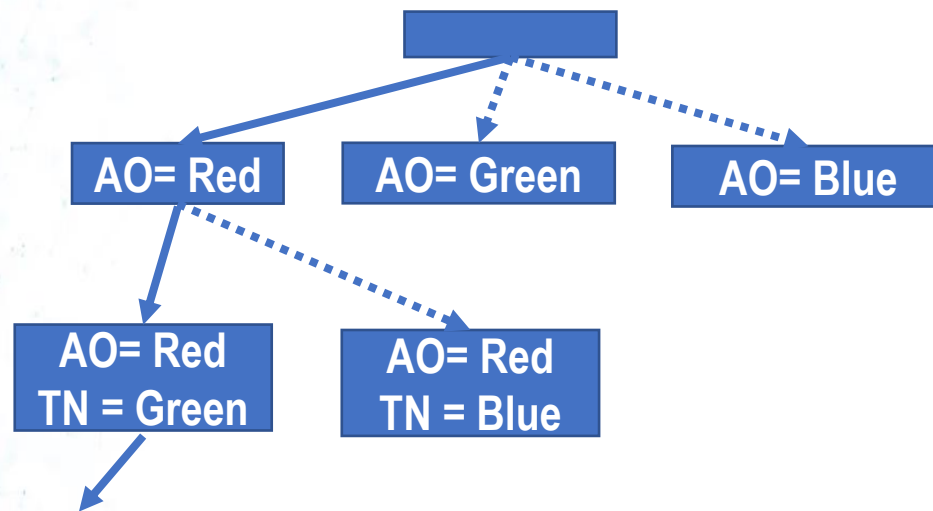
Entretanto, problemas CSP podem ser eficientes sem uso de heurísticas específicas do domínio de aplicação

- Com isso, busca em profundidade pode ser uma boa busca para ser usada em CSP

Resolvendo CSP: BUSCA c/ RETROCESSO

O que é Busca com Retrocesso ?

- É a Busca em Profundidade
- Atribui um valor para uma variável de cada vez (em cada nível)
- Faz backtracking quando não há valores válidos restantes para serem atribuídos
- Se tivermos n variáveis, a solução estará na profundidade n



Busca com Retrocesso










É uma busca cega. Podemos melhorá-la ?



Verificação Prévia (VP)

- Sempre que uma variável X tem valor atribuído
 - Verifica cada variável Y não-atribuída e conectada a X
 - Exclui do domínio de Y todos os valores conflitantes com o valor escolhido para X (forward checking)
 - VP é a **Busca por Retrocesso com Forward Checking**



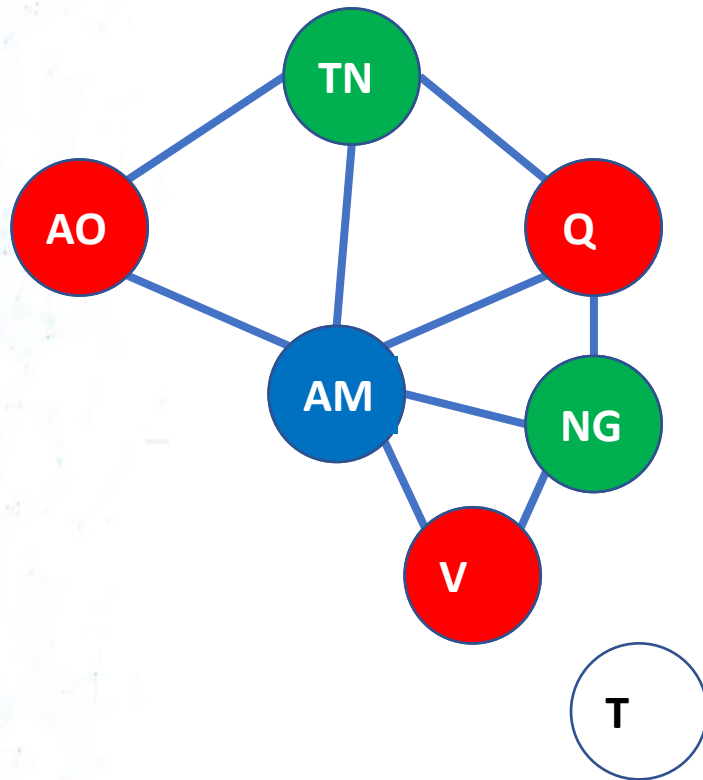
AO	TN	Q	NG	V	AM	T
						
						
						
						

Valores Restantes Mínimos (VRM)

Com a implementação da verificação prévia, podemos usar nossa primeira heurística genérica para um CSP → VRM

- VRM ou Variável mais restrita ou primeira falha
 - Escolher a variável que possui menos valores válidos
 - Ao aplicar VP, as variáveis não atribuídos vão ficando com cada vez menos valores válidos.
 - A variável com menos valores válidos possíveis é a mais restritiva e tem maior probabilidade de causar falha
- A escolha da variável de cada nível pode ser feita usando o VRM

Porque VRM acelera a busca ?



Se atribuirmos **RED** para **AO**

E **GREEN** para **TN**

Teríamos apenas um valor para
AM = Blue

A VRM força esta escolha

Com isso as demais escolhas se
Tornam todas forçadas

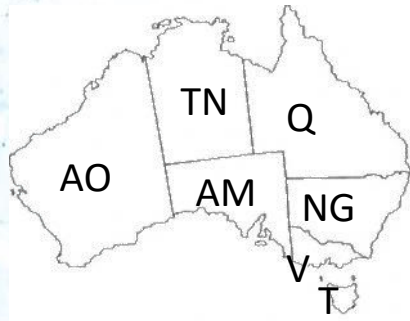
Podemos ajudar a VRM ?

Nossa segunda heurística aplicada a CSP é a:

- **Heurística de Grau**
- Perceba que VRM não ajuda muito na escolha da primeira variável no caso da Coloração de Mapas
 - Todas as variáveis possuem todos os valores como válidos.
- É aí que entra a heurística de grau:
 - Escolha a variável que possui o maior grau
 - Maior ligações com outro nós no grafo
 - Pode ser usada como critério de desempate para a VRM

É possível detectar caminhos falhos ?

VP detecta muitas inconsistências, mas não todas !

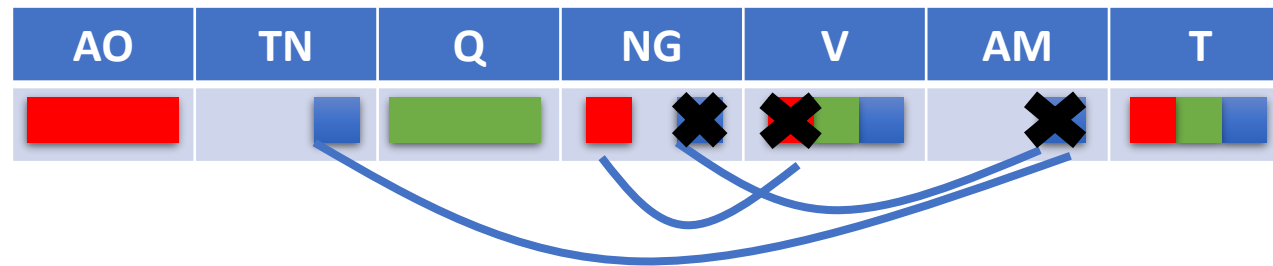


AO	TN	Q	NG	V	AM	T
Não percebe que TN e AM não podem ser BLUE						

- Tínhamos que propagar a restrição sobre AM e TN
 - Mas como fazer essa propagação de forma rápida ?

Consistência de Arco

- Verifique as variáveis repetidamente a cada atribuição eliminando inconsistências
- **2-consistência:** análise e compara 2 variáveis por vez



AM e NG serão consistentes se NG não for **BLUE**

NG e V serão consistentes se V não for **RED**

AM e TN serão consistentes se AM não for **BLUE**

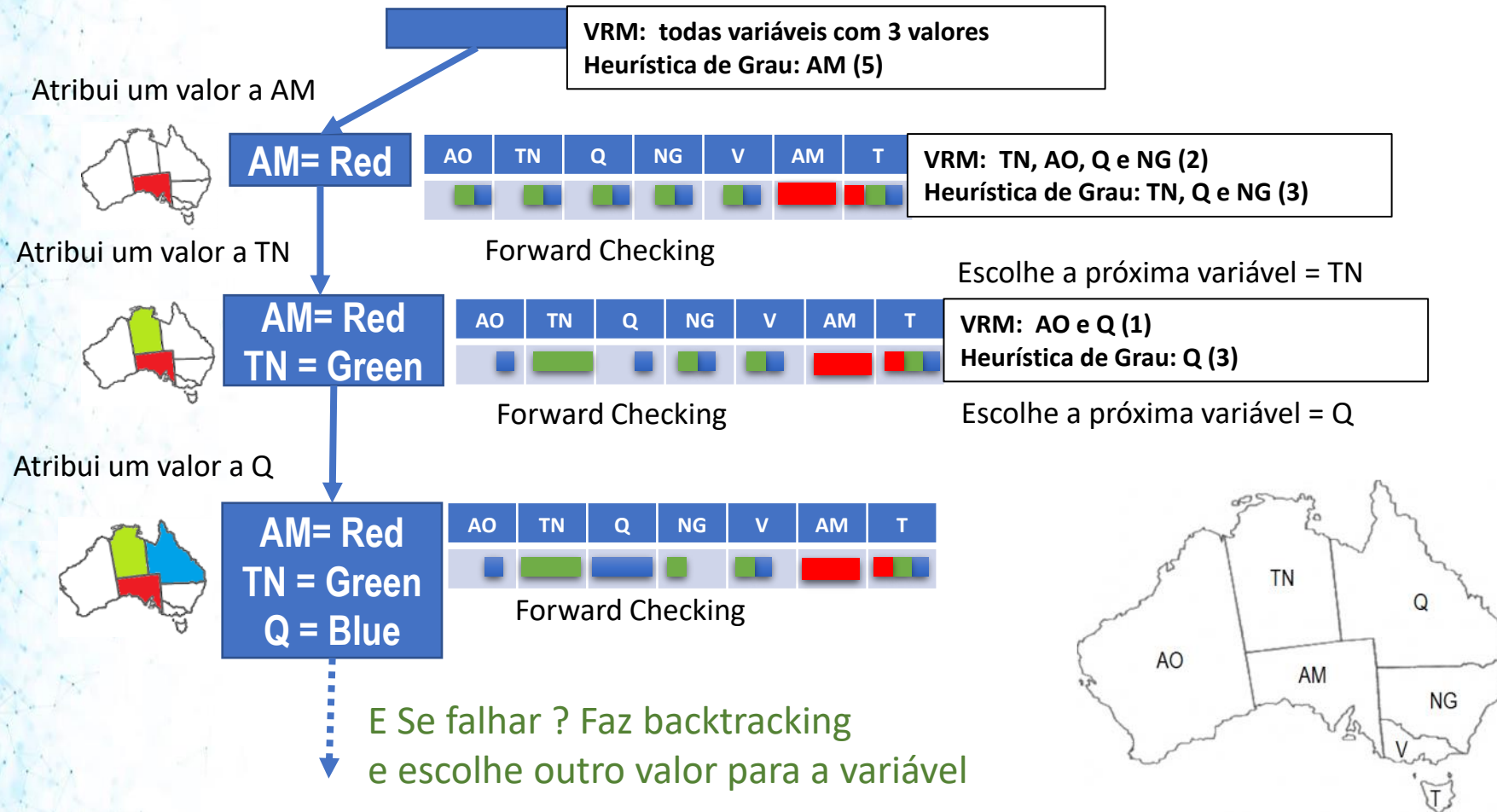
Detecta inconsistência antes da Verificação Prévia

Consistência de Arco

O custo é aproximadamente $O(n^2d^3)$ (p/ o CSP binário)

- n = número de variáveis
- d = valores possíveis para cada variável
- Como a Consistência de arco poda a árvore, este custo extra é compensador
- A Consistência de Arco garante a consistência entre pares de variáveis. Denomina-se **2-consistência**.
- Pode-se criar consistência entre grupos de K variáveis. Formando o que chamamos: **k-consistência**.
 - Aumento da complexidade do algoritmo de consistência
 - Determinar K depende de testes empíricos e cada CSP pode ter um k diferente

Busca Verificação Prévia com VRM e Heurística de Grau



Exemplos de CSP

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



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

- Aplicar:
 - Verificação Prévia ; VRM e k-consistência

Solucionando o Sudoku

Valorando as variáveis

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

Forward Checking

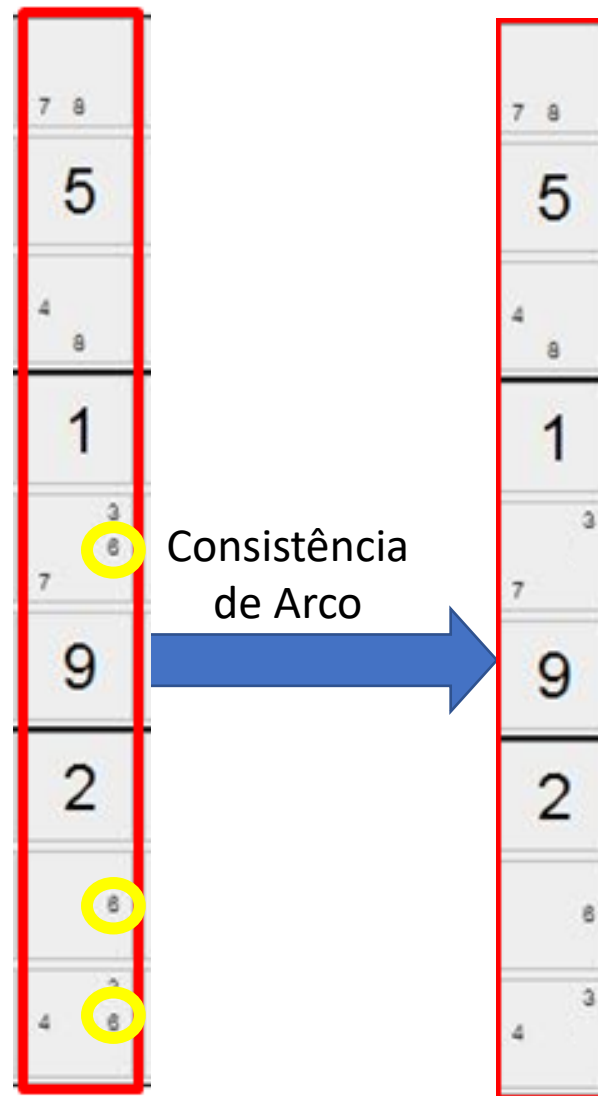


Busca Retrocesso (VP)
VRM + K-consistência

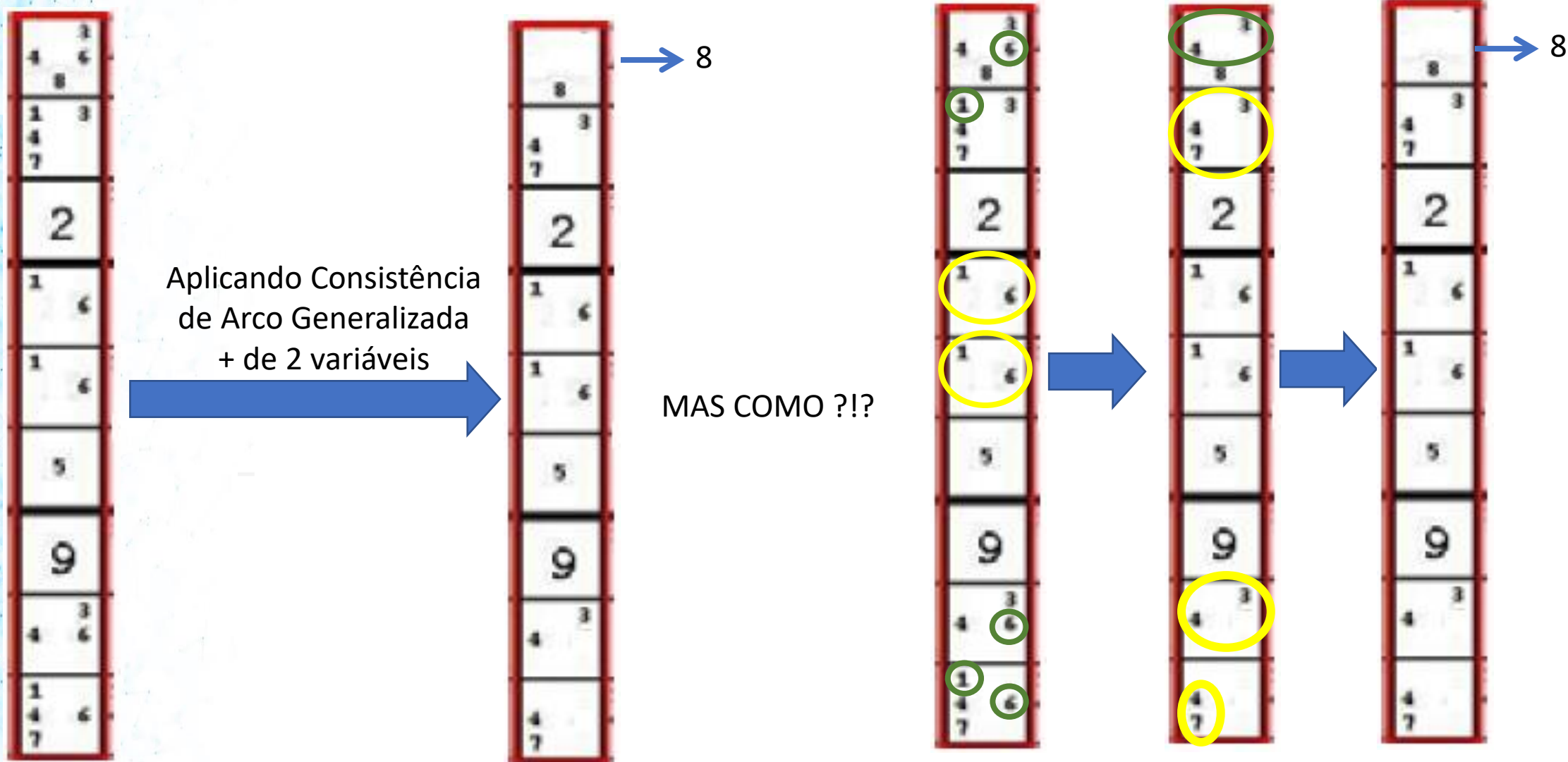
1 3 5 9	1 3 5 8 9	1 8 9	2 3 7 8 9	2 8 9	4	1 2 9	7 8	6
2	4 3 8 9	6	3 7 8 9	8 9	1	4 9	5	4 7 9
1 4	7	1 8 9	5	2 8 9	2 8 9	1 2 4	4 8	3
3 4 5 6 7 9	2 3 4 5 6 9	2 4	2 8 9	2 5 6 9	2 5 6 9	8	1	2 4 5 7
8	1 2 3 5 6 9	1 2 7 9	1 2 8 9	4 5 6 9	2 5 6 9	2 3 5 6 7	3 8	2 5 7
1 4 5 6	1 2 4 5 6	1 2 4	1 2 8	3 8	7	2 4 5 6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

Solucionando o Sudoku

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



Solucionando o Sudoku



Busca Local para CSP

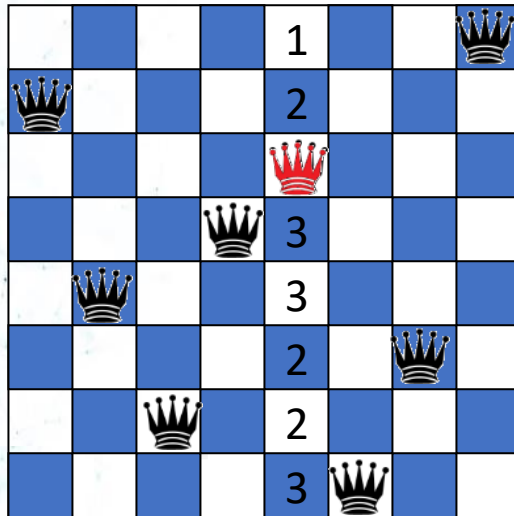
O que é uma Busca Local ?

Exemplo de Busca Local: **Subida da Encosta**

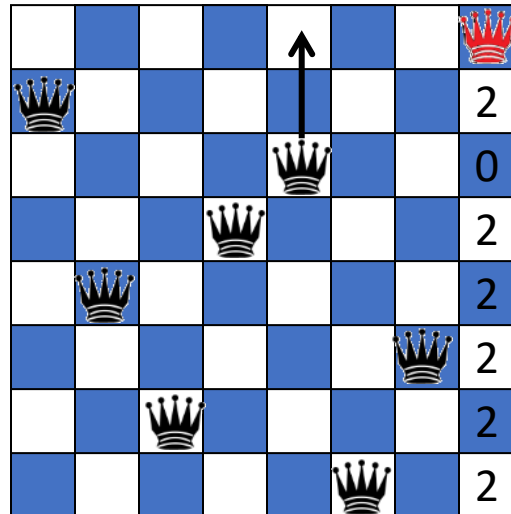
Característica da Busca Local para CSP

- Usa uma representação completa de um estado
 - Permitindo estado com restrições não satisfeitas
 - A mudança de nó (estado) se dá pela re-atribuição de valores das variáveis
- Seleção de variáveis: **aleatório pra qualquer variável em conflito**
- Seleção do valor da variável: **heurística do mínimo conflito**
 - Selecione o novo valor que resulta no mínimo número de conflitos com outras

Busca Conflitos Mínimos



Jogo das 8-Rainhas
Configuração inicial



1. Escolha uma variável em conflito (aleatório)
2. Atribua o valor de conflito mínimo
3. Repita o processo...
4. ... Até acabar os conflitos

Busca Local com Conflitos Mínimos

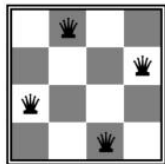
- Tem como critérios de parada:
 - Número máximo de iterações (re-atribuição de valores de variáveis)
 - Encontrar um estado sem conflitos
- É muito útil e resolve vários CSP
 - Resolve n-rainhas muito rapidamente
 - Motivo: Soluções são densamente distribuídas na matriz e a solução está a pouca distância do estado inicial aleatório
- Esta busca é empregada no telescópio Hubble
 - Reduz o tempo de programação de observações de 3 semanas para apenas 10 minutos

Comparativo dos métodos e heurísticas

Vimos 3 processos de Busca: Retrocesso , VP (retrocesso + Forward Checking) e Conflitos Mínimos

Vimos 2 heurísticas: VRM e Heurística de Grau

Quantidade de verificações de consistência



Problema	Retrocesso (simples)	Retr+VRM	Verificação Prévia	VP+VRM	Conflitos Mínimos
Colorir EUA	> 1.000 mil sem solução	> 1.000 mil s/ solução	2 mil	60	64
n-rainhas n = 2 a 50	> 40.000 mil sem solução	13.500 mil	>40.000 mil sem solução	817 mil	4 mil

Aplicações

- CSP permite solucionar problemas complexos:
 - Atribuição de tripulação para vôos
 - Gerenciamento de frota de transporte
 - Escalonamento de vôos
 - Alguns casos de escalonamento de tarefas
 - Design
 - Cirurgia de Cérebros
 - Escalonamento de Aulas e Professores

Exercício: Criptoaritmética

Como modelar o problema abaixo em CSP ?

$$\begin{array}{r} \text{ T W O} \\ + \text{ T W O} \\ \hline \text{ F O U R} \end{array}$$

Exercício: Criptoaritmética

Como modelar o problema abaixo em CSP ?

$$\begin{array}{r} \overset{x_3}{T} \overset{x_2}{W} \overset{x_1}{O} \\ + T \\ \hline F U \end{array}$$

- **Variables:** $F T U W R O$ $X_1 X_2 X_3$
- **Domains:** $\{0,1,2,3,4,5,6,7,8,9\}$ $\{0,1\}$
- **Constraints:** $AllDif \{F,T,U,W,R,O\}$
 - $O + O = R + 10 \cdot X_1$
 - $X_1 + W + W = U + 10 \cdot X_2$
 - $X_2 + T + T = O + 10 \cdot X_3$
 - $X_3 = F, T \neq 0, F \neq 0$

Bibliografia desta Aula

aprofundamento nos assuntos desta aula, segue a seguinte referência bibliográfica

- Russel & Norvig (Artificial Intelligence)
 - Capítulo 5
- Alguns slides desta aula foram baseados no slides:
- Lise Getoor – CMSC 421 – Fall 2006 – Slides. University of Mariland, 2006