Problemas de satisfação de restrições: definição; inferência

definição

Problema de Satisfação de Restrições *aka*

*Constraint Satisfaction Problem (**CSP**)*

representação fatorizada em variáveis

problema fica resolvido assim que os valores das variáveis satisfizerem todas as restrições

situando o tema

- planeamento: sequência de ações
  - o percurso até ao objetivo é o importante
  - os percursos têm custos e profundidades diferentes

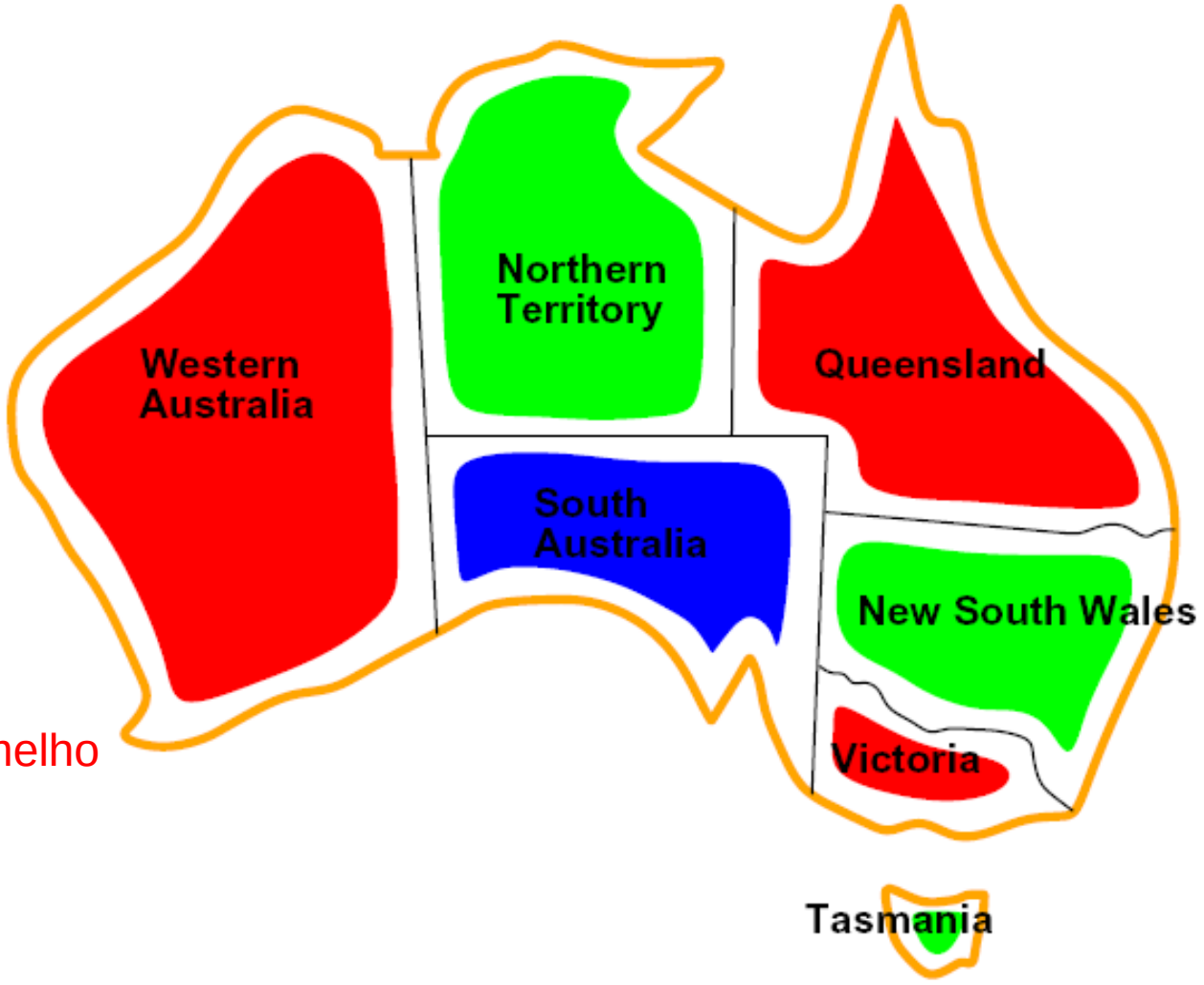
- **identificação**: atribuição de valores a variáveis
  - o objetivo é o que importa, não o percurso
  - todos os percursos têm o mesmo custo e profundidade (em geral)
  - CSPs são uma classe específica de problemas de identificação

exemplos

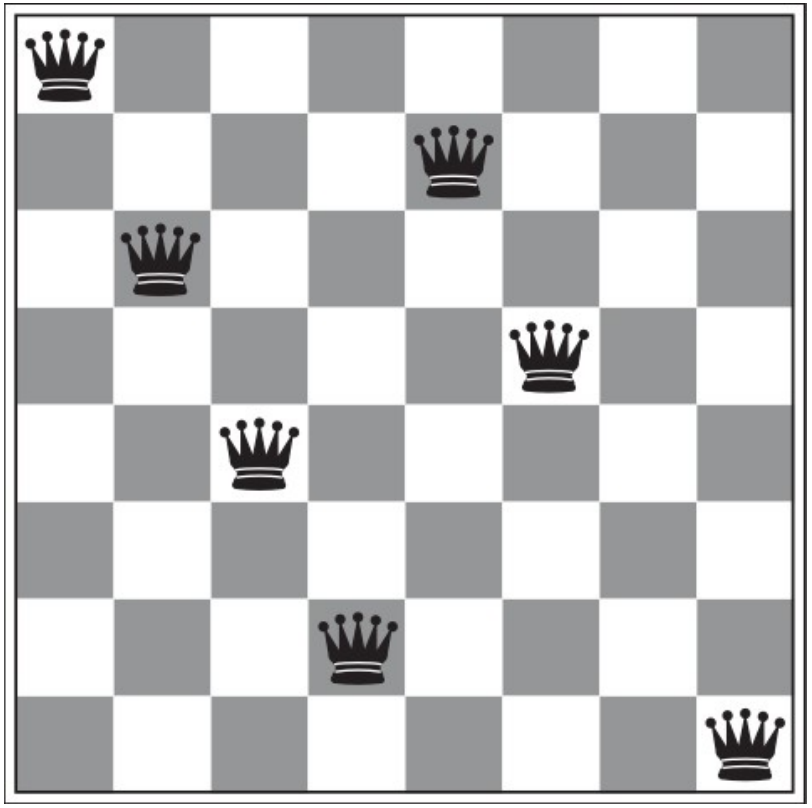
problema da coloração de um mapa

como pintar um mapa de modo a que regiões adjacentes tenham cores diferentes?

ex: Austrália com azul, verde e vermelho



problema das n-rainhas



formalizando um CSP

X: conjunto das variáveis, $\{X_1, ..., X_n\}$

D: conjunto dos domínios, $\{D_1, ..., D_n\}$, um para cada variável

C: conjunto de restrições que especifica combinações de valores admissíveis

formalizando

D_i: conjunto de valores possíveis, $\{v_1, ..., v_k\}$ para a variável X_i

C_i: *<âmbito, rel>*

âmbito: tuplo de variáveis envolvidas na restrição *i*

rel: relação que define os valores que as variáveis podem tomar

formalizando a coloração do mapa

variáveis

$X = \{WA, NT, Q, NSW, V, SA, T\}$

domínios

$D = \{azul, verde, vermelho\}$

restrições, ex:

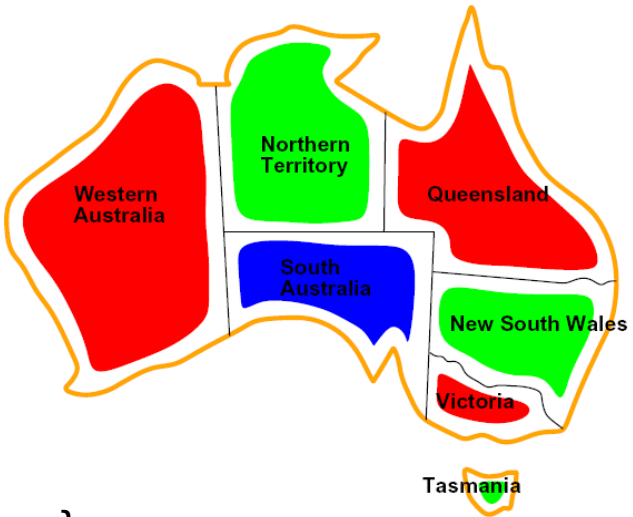
implícita: $WA \neq NT$

explícita: $(WA, NT) \in \{(verde, vermelho), (azul, vermelho), \ldots\}$

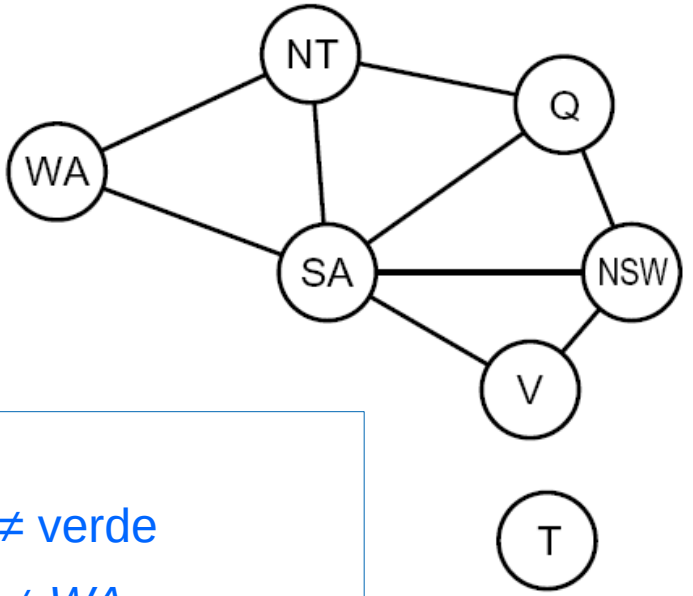
soluções

são atribuições (há várias) que satisfazem todas as restrições, ex:

{*WA*=verm, *NT*=verde, *Q*=verm, *NSW*=verde, *V*=verm, *SA*=azul, *T*=verde}



CSP como um grafo



CSP binário

cada restrição relaciona (no máximo) duas variáveis

grafo de restrições binárias

nós são variáveis
arcos são restriçõess

restrições

unárias: *SA* ≠ verde

binárias: *SA* ≠ *WA*

ordem superior: nº arbitrário de variáveis

algoritmos gerais de CSP

usam a estrutura do grafo para acelerar a procura
(no ex., Tasmânia é um subproblema independente)

formalizando as rainhas v1

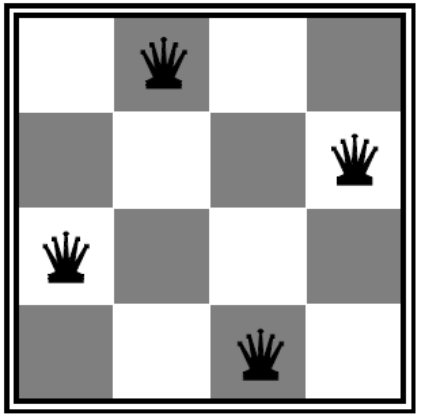
- formulação 1

  variáveis: X_{ij} (casas do tabuleiro)

  domínios: $\{0,1\}$

  restrições:

$$\forall i,j,k \quad (X_{ij}, X_{ik}) \in \{(0,0),(1,0),(0,1)\}$$
$$\forall i,j,k \quad (X_{ij}, X_{kj}) \in \{(0,0),(1,0),(0,1)\}$$
$$\forall i,j,k \quad (X_{ij}, X_{i\pm k, j\pm k}) \in \{(0,0),(1,0),(0,1)\}$$
$$\sum_{i,j} X_{i,j} = N$$



formalizando as rainhas v2

- formulação 2

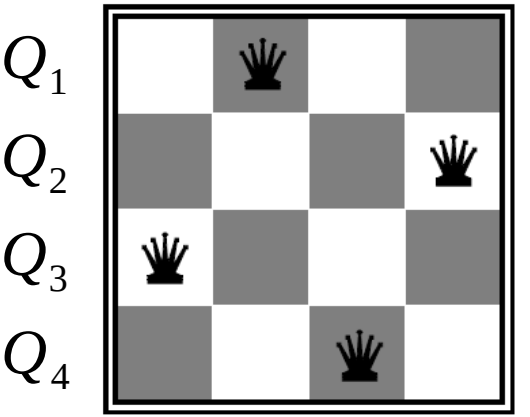
  variáveis: Q_k (rainha da linha *k*)

  domínios: $\{1, 2, \ldots, N\}$

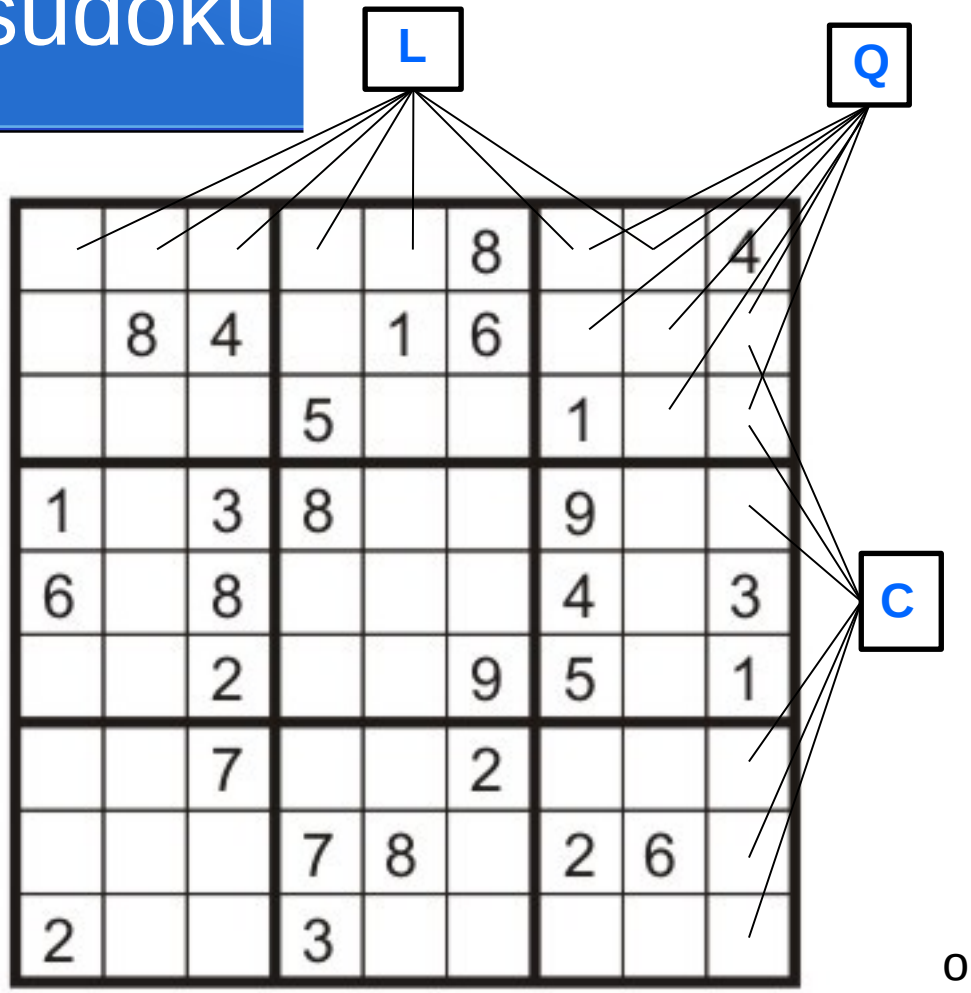
  restrições:

  implícitas: $\forall i, j$ não-ameaçadas(Q_i, Q_j)

  explícitas: $(Q_1, Q_2) \in \{(1,3)(1,4), \ldots\}$



sudoku



restrições de ordem superior (globais):
nº arbitrário de variáveis
não necessariamente todas

variáveis: cada casa vazia

domínios: {1, 2, …, 9}

restrições: ***alldiff***

- 9 casas *todas diferentes* em cada coluna **C**
- 9 casas *todas diferentes* em cada linha **L**
- 9 casas *todas diferentes* em cada quadrado **Q**

ou imensas desigualdades binárias

inferência ou propagação de restrições

usar as restrições para reduzir o domínio de uma variável

o que pode reduzir o domínio de outra, etc.

é um passo prévio à procura

mas também pode voltar a fazer-se ao longo da procura

- resultado:
  - várias soluções possíveis
  - uma solução possível
  - nenhuma solução possível

há casos em que a propagação resolve o problema sem procura!

propagação de restrições

variável nó consistente

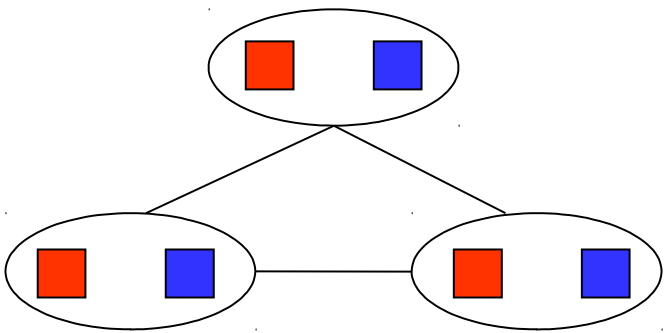
todos os valores do domínio da variável satisfazem as suas restrições unárias

se isso não acontecer, basta reduzir o domínio da variável

variável arco consistente

todos os valores do domínio da variável satisfazem as suas restrições binárias

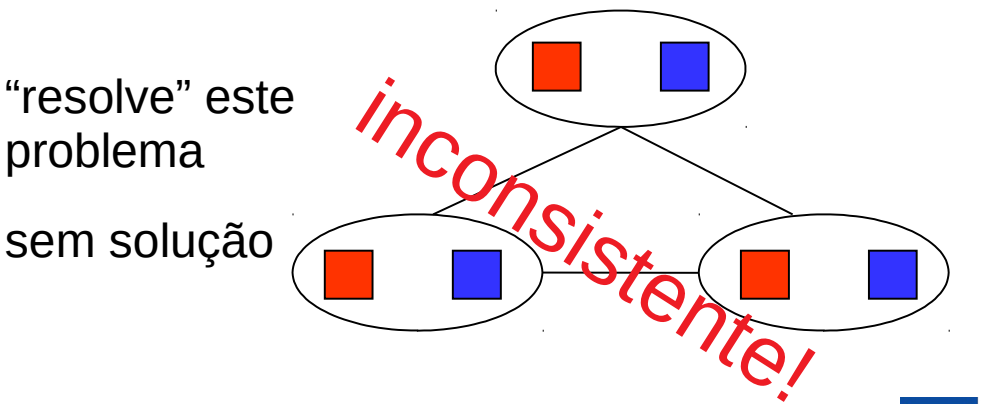
ou: para qualquer valor de uma variável é sempre possível escolher um valor da outra variável



hélas!

consistência de percurso

$\{X_i, X_j\}$ são consistentes em percurso relativamente a X_m se para cada $\{X_i = a, X_j = b\}$ consistente com as restrições de $\{X_i, X_j\}$ existir uma atribuição a X_m que satisfaz as restrições de $\{X_i, X_m\}$ e $\{X_m, X_j\}$



k-consistente

caso geral de consistência entre *k* variáveis

- *k* = 1: consistência de nó
- *k* = 2: consistência de arco
- *k* = 3: consistência de percurso (em grafos binários)

se o problema for fortemente *k*-consistente

- também é (*k-1*)-consistente, (*k-2*)-consistente, …, 1-consistente

hélas!

- tem complexidade temporal **e espacial** exponencial em *n*

nº de nós

restrições de ordem superior

com nº arbitrário de variáveis

alldiff – deteção simples

com *m* variáveis e *n* valores diferentes possíveis,

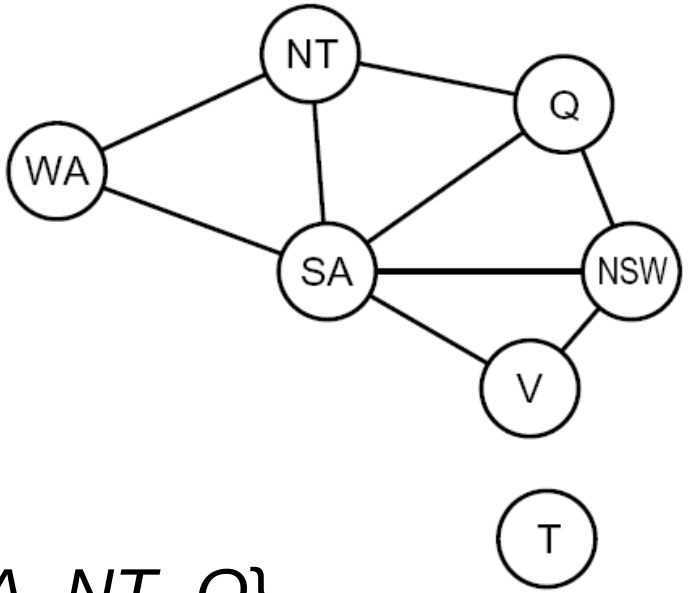
se *m* > *n*, a restrição não é satisfeita

alldiff – algoritmo simples

1. remover uma variável com domínio singular e remover esse valor dos domínios da outras variáveis
2. repetir enquanto houver variáveis singulares

   se surgir um domínio vazio, ou *m* > *n*, então há uma inconsistência

exemplo *alldiff*



suponhamos a atribuição
{*WA* = vermelho, *NSW* = vermelho}

do grafo (binário) observa-se *alldiff* {*SA*, *NT*, *Q*}

mas como *SA*, *NT* e *Q* não podem ser vermelho, só restam duas cores (azul e verde) para 3 variáveis ($m > n$)

⇒ *alldiff* {*SA*, *NT*, *Q*} é violada

restrição global pode ser mais eficiente do que consist. arco

generalidade das estratégias CSP

após definir um problema em termos de restrições

as estratégias

- consistência de arco
- consistência de percurso
- *alldiff*
- …

aplicam-se a qualquer CSP

- (sudoku, escalonamento de tarefas numa fábrica, ...)

para quem quiser explorar mais...

- restrições de recursos, *atmost*

  ex: *atmost* {10, X_1, X_2, X_3, X_4}, soma de X_1, …, $X_4 \leq 10$

- domínios como intervalos e **propagação de intervalos**

  ⇒ consistência de intervalos

- restrições de **preferência**

  ⇒ **problema de otimização de restrições**

- domínios infinitos (discretos)

  ⇒ linguagem de restrições

- domínios contínuos

  programação linear

exercício

- experimentar consistência de arco

  se resolver todo, experimentar num sudoku mais difícil

- experimentar com consistência de caminho

  num sudoku mais difícil

- e experimentar com *alldiff*

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