Problemas de Satisfação de Restrições

Texto base:

Stuart Russel e Peter Norving - "Inteligência Artificial"
David Poole, Alan Mackworth e Randy Goebel -
"*Computational Intelligence – A logical approach*"

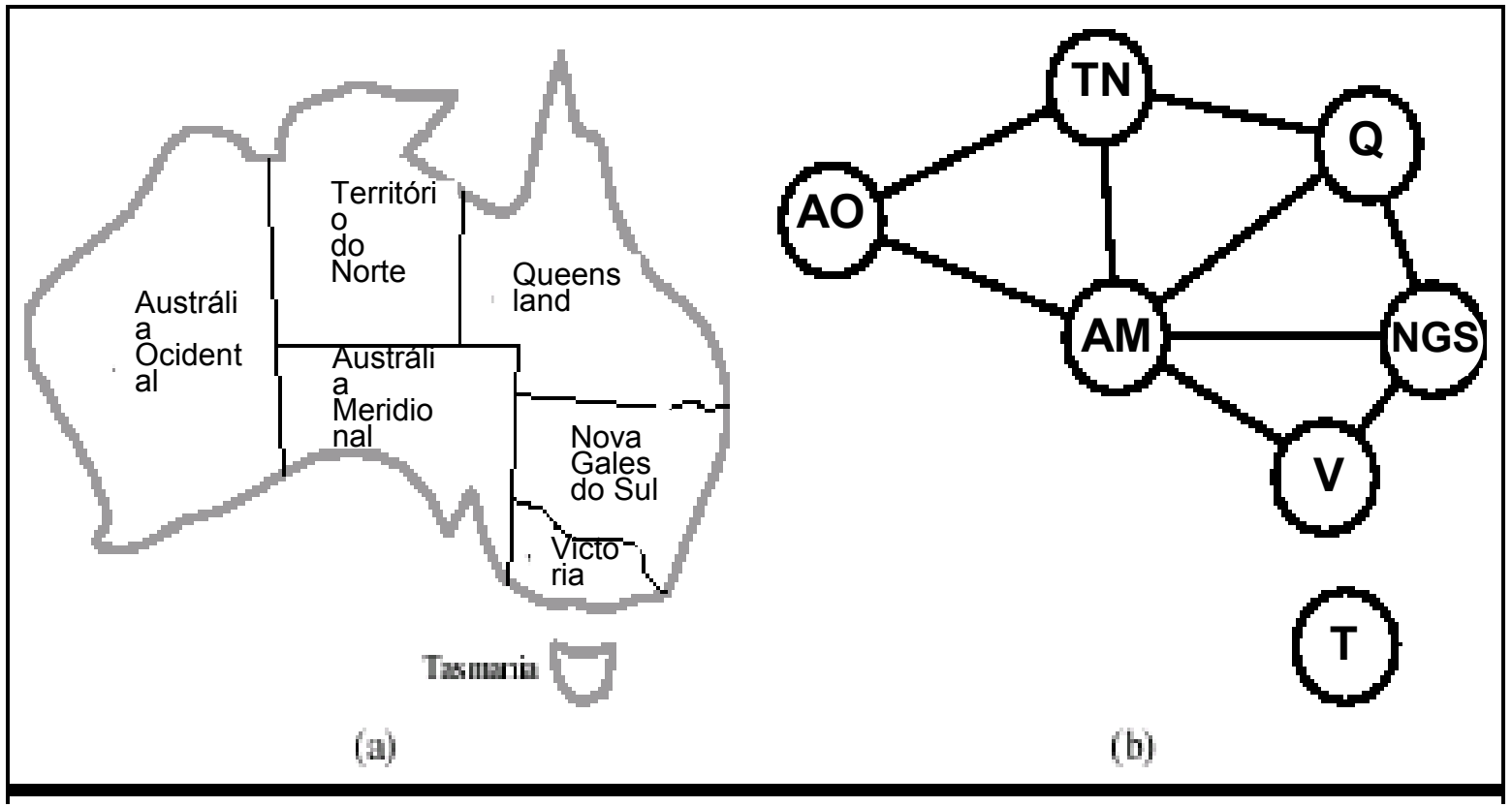
junho/2007

O que é um Problema de Satisfação de Restrição (PSR)?

- Um PSR consiste em:
  - Um **conjunto de variáveis** que podem assumir valores dentro de um dado domínio
  - Um **conjunto de restrições** que especificam propriedades da solução - valores que essas variáveis *podem* assumir ou
  - Uma **função de avaliação** que especifica o quão boa é um conjunto de atribuições de valores para variáveis
  - Um **estado** do problema é definido por uma atribuição de valores a alguma ou a todas as variáveis
  - Atribuições:
    - Consistente ou válida: que não viola nehuma restrição
    - Completa: todas as variáveis são atribuídas
  - Solução: atribuição completa que satisfaz todas as restrições

Exemplo: coloração de Mapas

- Queremos colorir cada região do mapa abaixo com as cores vermelho, verde e azul
- Nenhuma região vizinha deve ter a mesma cor



Problema da coloração de mapas como um PSR

- Variáveis - Cada uma das regiões:AO, TN, Q, NGS, V, AM e T
- Domínio das variáveis: {vermelho, verde, azul} para todas
- Restrições: regiões vizinhas devem ter cores distintas
  - Exemplo: combinações permissíveis para AO e TN
    - {(vermelho, verde), (vermelho, azul), (verde, vermelho), (verde, azul), (azul, vermelho), (azul, verde)}
    - Ou simplesmente AO ≠ TN

Características das restrições

- O conjunto de valores que a variável *i* pode assumir é chamado **domínio** D_i que pode ser:
  - **discreto –** ex: fabricantes de uma peça de carro
  - **contínuo** – ex: peso das peças do carro
- Quanto à aridade, as **restrições** podem ser:
  - **Unárias** (única variável)- ex: a Tasmânia não pode ser verde
  - **Binárias** (duas variáveis) - ex.: 8-rainhas, coloração mapas
  - **N-árias** - ex.: palavras cruzadas
- Quanto à natureza, as **restrições** podem ser:
  - **Absolutas** (não podem ser violadas) – ex: AO ≠ TN
  - **Preferenciais** (devem ser satisfeitas quando possível) – ex: problema de elaboração de horários

PSRs podem ser divididos em duas classes de principais de problemas

- Problemas de satisfatibilidade
  - O objetivo é encontrar uma associação de valores para as variáveis que satisfaça algumas restrições
  - Exemplos: Criptoaritmética, n-rainhas
- Problemas de otimização
  - Cada associação de um valor para cada variável tem um custo
  - O objetivo é encontrar uma associação com o menor custo (associação ótima)

Relacionamento com Busca

- O caminho até o objetivo não é importante, somente a solução
- Muitos algoritmos exploram a natureza multidimensional dos problemas
- Não existe um nó inicial pré-definido
- Frequentemente estes problemas são grandes, com centenas de variáveis
  - Explorar sistematicamente o espaço de estados pode ser inviável
- Para problemas de otimização não existem nós objetivos bem definidos

Benefícios de se tratar um problema de busca como um PSR

- Representação de estados obedece um padrão definido
  - Conjunto de variáveis com valores atribuídos
- Função sucessor e teste de objetivo podem ser definidos de forma genérica para todos os PSRs
- Heurísticas efetivas e genéricas podem ser desenvolvidas sem nenhum conhecimento sobre o domínio dos problemas
- A estrutura do grafo de restrições pode ser utilizada para reduzir a complexidade do problema

Exemplo de PSR para escalonamento de atividades

- Variáveis: A, B, C, D e E que representam o tempo de início das várias atividades
- Domínios: $D_A = D_B = D_C = D_D = D_E = \{1, 2, 3, 4 , 5\}$
- Restrições: $(B \neq 3) \wedge (C \neq 2) \wedge (A \neq B) \wedge (B \neq C) \wedge (B \neq D) \wedge (C < D) \wedge (A = D) \wedge (E < A) \wedge (E < B) \wedge (E < C) \wedge (E < D)$

Algoritmos gerar-e-testar

- Gerar o espaço de associações D
  - $D = D_{V1} \times D_{V2} \times D_{V3} \times \ldots \times D_{Vn}$
- Testar cada uma das associações com as restrições
- Exemplo:
  - $D = D_A \times D_B \times D_C \times D_D \times D_E$
  - $D = \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$
  - $D = \{<1, 1, 1, 1, 1>, <1, 1, 1, 1, 2>, <1, 1, 1, 1, 3>, \ldots <4, 4, 4, 4, 4>\}$
- Gerar e testar é sempre exponencial
  - *n* variáveis com domínios de tamanho *d* e *e* restrições – $O(ed^n)$ testes

Algoritmos com retrocesso

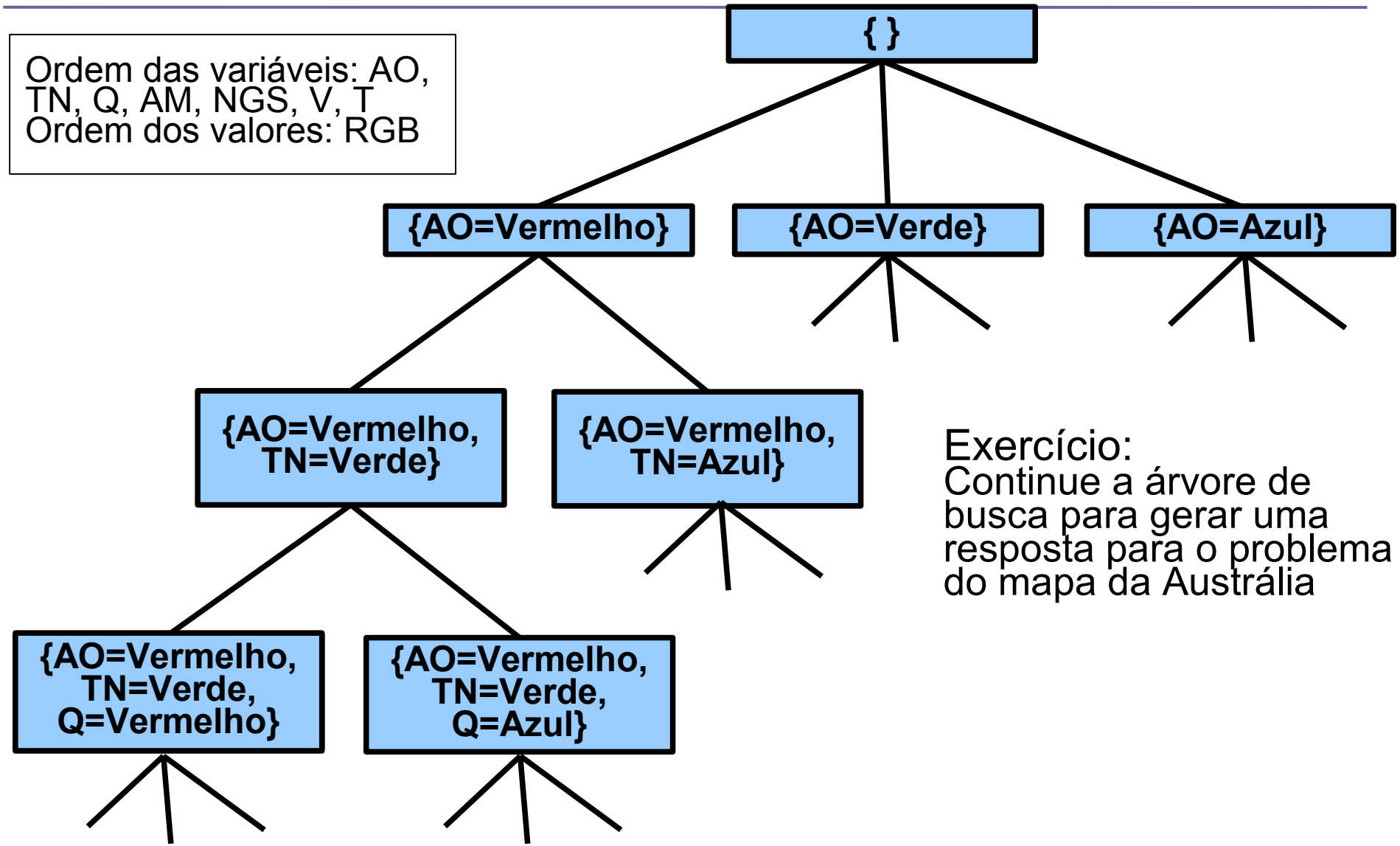
- Explorar D sistematicamente
  - Associar as variáveis aos seus valores em alguma ordem e avaliar as restrições que envolvem as variáveis já atribuídas
  - Qualquer associação que não satisfaça as restrições pode ser podada
- Exemplo: A associar A=1 e B=1 viola a restrição A≠B, independente dos valores das outras variáveis

PSR como uma busca em grafo

- Ordenar as variáveis V_1, V_2, ..., V_n
- O nó inicial é uma associação vazia {}
- Um nó associa valores para as *j* primeiras variáveis
- Os vizinhos do nó $\{V_1/v_1, V_2/v_2, ..., V_j/v_j\}$ são todos os nós consistentes $\{V_1/v_1, V_2/v_2, ..., V_j/v_j, V_{j+1}/v_{j+1}\}$ para cada valor v_{j+1} *no domínio de* V_{j+1}
- O nó objetivo é um associação completa e válida (que satisfaz todas as restrições)
- Podemos resolver o problema usando busca em profundidade
  - As soluções apareceram sempre no profundidade = ao número de variáveis envolvidas

Parte da árvore de busca gerada pela busca em profundidade

Ordem das variáveis: AO, TN, Q, AM, NGS, V, T
Ordem dos valores: RGB



Exercício:
Continue a árvore de busca para gerar uma resposta para o problema do mapa da Austrália

Ordenando variáveis e valores

- A idéia é tentar gerar uma falha logo no início da busca, podando os galhos que não terão sucesso
- Em que ordem testar as variáveis?
  - Em ordem crescente do tamanho do domínio (heurística de variável mais restrita ou valores restantes mínimos)
  - Em ordem decrescente pela quantidade de restrições que cada variável está envolvida (heurística de grau)
- Em que ordem testar os valores?
  - Valores que restringem menos as atribuições futuras devem ser testados antes (heurística de valor menos restritivo)
  - Eliminar dos domínios das variáveis seguintes os valores que não poderão mais serem utilizados devido às atribuições anteriores (verificação prévia)

Exemplo para o mapa da Austrália

	AM (5)	TN (3)	Q (3)	NGS (3)	AO (2)	V(2)	T (0)
DOMÍNIO	RGB	RGB	RGB	RGB	RGB	RGB	RGB
AM = R	**R**	GB	GB	GB	GB	GB	RGB
TN = G		**G**	B	GB	B	GB	RGB
Q = B			**B**	G	B	GB	RGB
NGS = G				**G**	B	B	RGB
AO = B					**B**	B	RGB
V = B						**B**	RGB
T = R							**R**

Heurísticas Gerais para PSRs

- São independentes do domínio do problema
- Podemos encontrá-las resolvendo as seguintes questões:
  - Que variável deve ser atribuída em seguida?
  - Em que ordem os valores devem ser experimentados?
  - Quais são as implicações das atribuições de variáveis atuais para as outras variáveis não atribuídas?
  - Quando um caminho falha*, a busca pode evitar repetir esta falha em caminhos subseqüentes?

* quando alcançamos um estado em que uma variável não tem nenhum valor válido

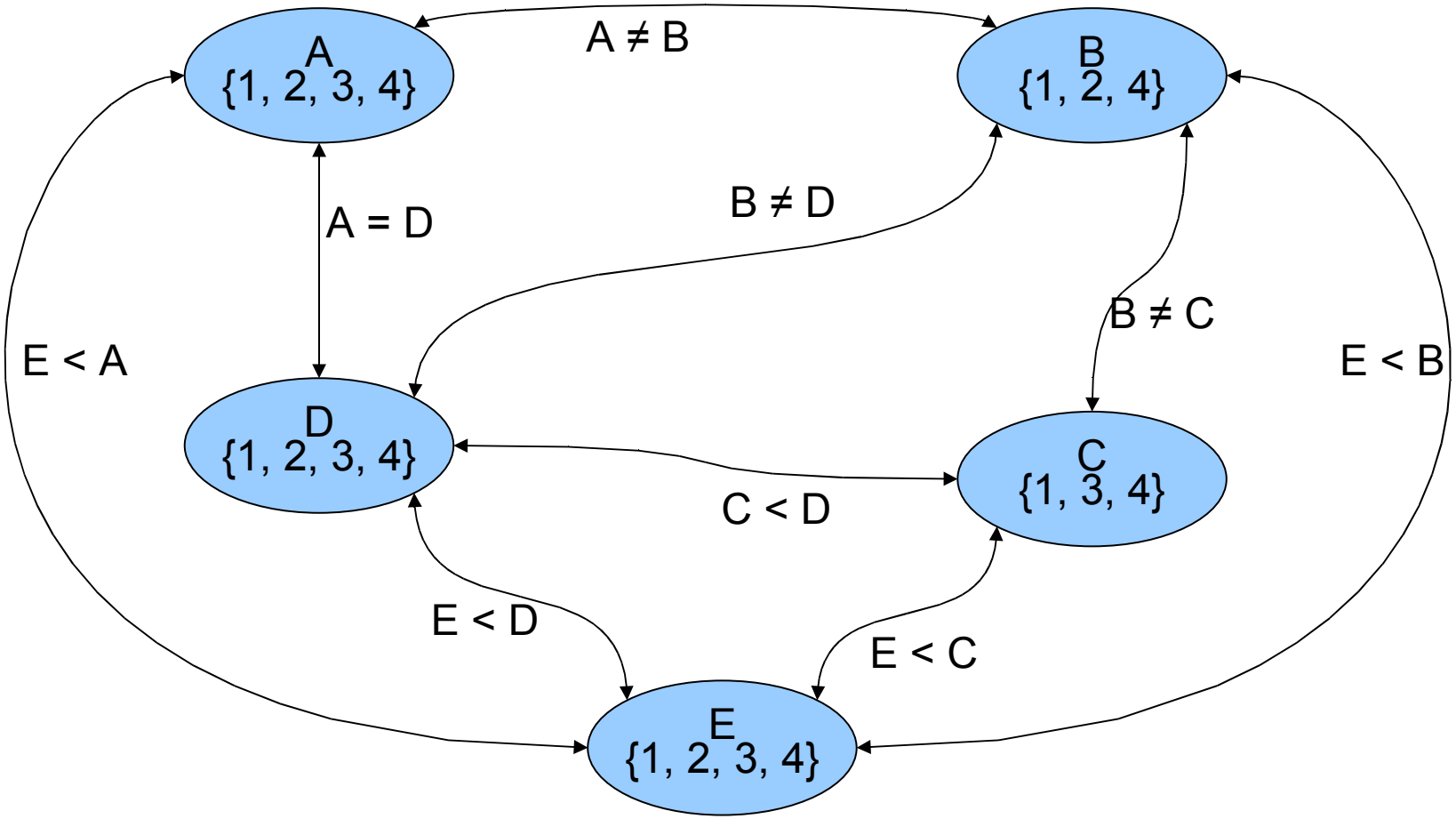
Algoritmos de consistência

- Idéia: podar os domínios tão logo quanto possível, antes de que seus valores sejam selecionados
- Uma variável é **domínio consistente** se nenhum valor do domínio é impossível por qualquer uma das restrições
- Exemplo DB = {1, 2, 3, 4} não é domínio consistente porque B=3 viola a restrição B≠3

Consistência de arcos

- Uma rede de restrições tem nós correspondentes às variáveis e seus domínios associados
  - Cada relação de restrição P(X, Y), corresponde aos arcos <X,Y> e <Y,X>
- Um arco <X,Y> é um arco consistente se para cada valor de X em D_X, existe algum valor de Y em D_y, para o qual a restrição P(X,Y) é satisfeita
  - Uma rede é arco consistente se todos os seus arcos são arco consistentes

Uma rede de restrições para o problema de escalonamento de atividades



Algoritmo de consistência de arcos AC-3

- Se um arco <X,Y> *não* é arco consistente, todos os valore de X em D_X para os quais não existe nenhum valor correspondente em D_y podem ser apagados de D_X para fazer <X,Y> arco consistente
- Os arcos são considerados um a um tornando cada um deles consistente
- Um arco <X,Y> necessita ser revisado se o domínio de Y for reduzido