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*"

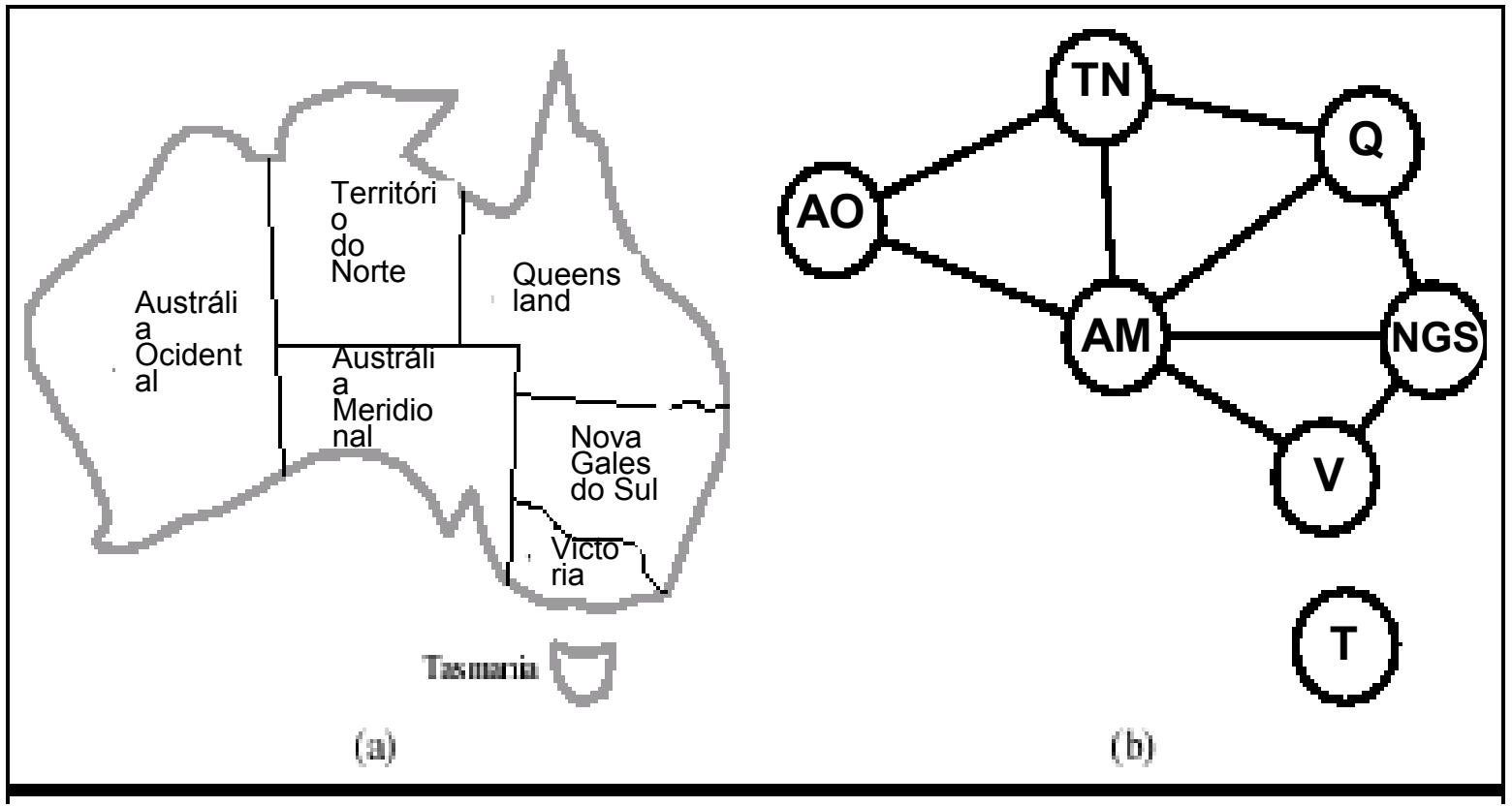
julho/2008

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

- Um PSR consiste em:
  - Um **conjunto de variáveis** que podem assumir um **conjunto de valores** dentro de um dado domínio
  - Um **conjunto de restrições** que especificam propriedades da solução - valores que essas variáveis *podem* assumir
  - Pode existir uma **função de avaliação** que especifica o quão boa é um conjunto de atribuições de valores para as 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: uma atribuição completa e válida

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
  - AO≠AM, TN≠AM, Q≠AM, NGS≠AM, V≠AM, AO≠TN, TN≠Q, Q≠NGS, NGS≠V
  - 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 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, coloração de mapas
- 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)
  - Exemplo: Elaboração de horário de aula

PSRs e o Relacionamento com Busca

- O caminho até o objetivo não é importante, somente a solução
- 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
  - Função sucessor: atribuir um valor válido a uma variável
  - Teste de objetivo: todas as variáveis com valores atribuídos
- 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

Algoritmos gerar-e-testar

- Gerar o espaço de associações D
  - $D = D_{AO} \times D_{TN} \times D_{Q} \times \ldots \times D_{T}$
- Testar cada uma das associações com as restrições
- Exemplo:
  - $D = D_{AO} \times D_{TN} \times D_{Q} \times D_{NGS} \times D_{V} \times D_{AM} \times D_{T}$
  - $D = \{R,G,B\} \times \{R,G,B\} \times \{R,G,B\} \times \{R,G,B\} \times \{R,G,B\} \times \{R,G,B\} \times \{R,G,B\}$
  - D = {<R,R,R,R,R,R,R>, <R,R,R,R,R,R,G>, <R,R,R,R,R,R,B>, ... <B,B,B,B,B,B,B>}
- Gerar e testar é sempre exponencial
  - *n* variáveis com domínios de tamanho *d* e *e* restrições – $O(ed^n)$ testes

Algoritmos de busca com retrocesso

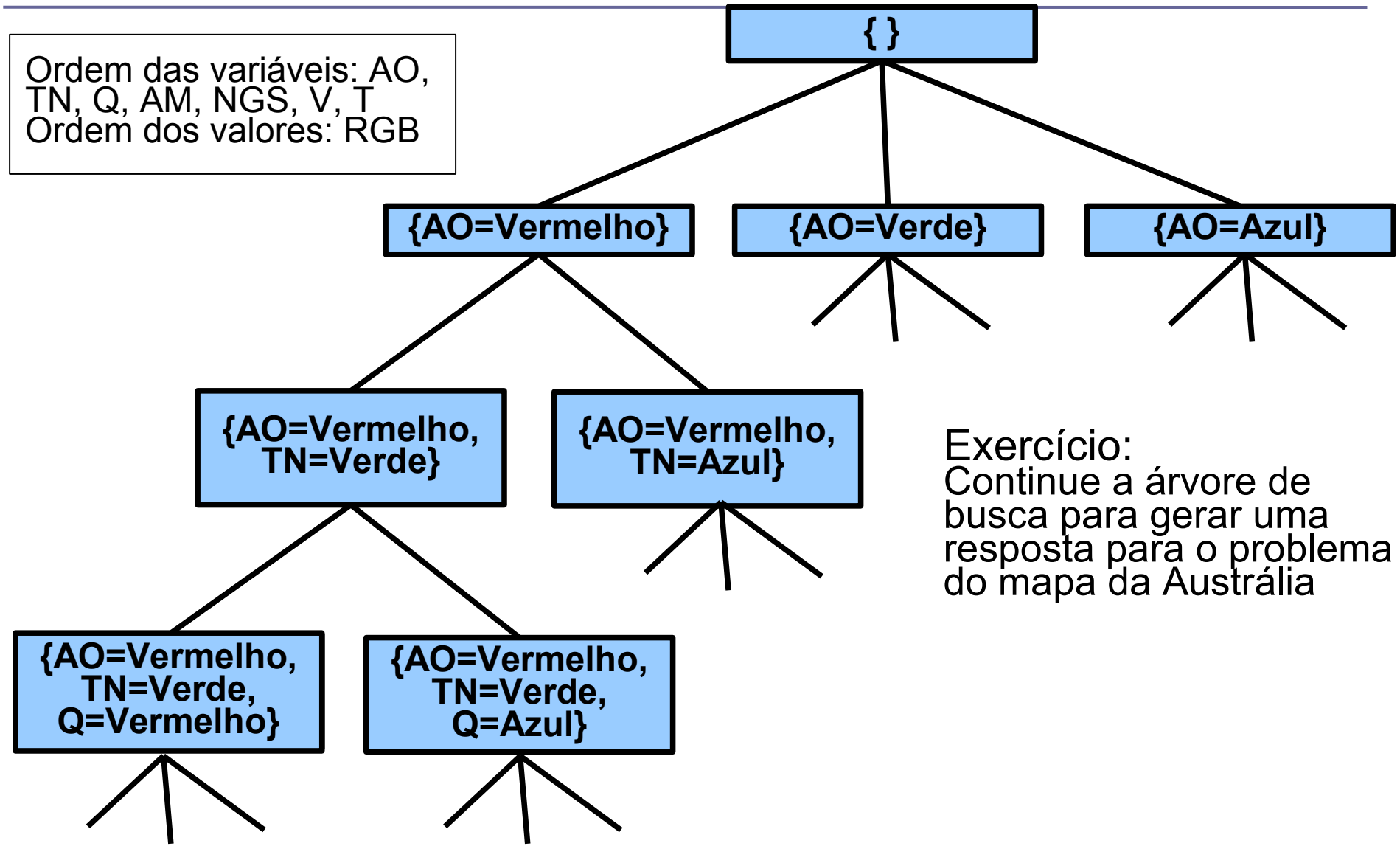
- Explorar o espaço de associações D sistematicamente
  - Associar as variáveis aos seus valores em alguma ordem e avaliar as restrições que envolvem as variáveis já atribuídas
  - Qualquer associação que não satisfaça as restrições pode ser podada
- Exemplo: associar AO=R e TN=R viola a restrição AO≠TN,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

PSR como uma busca em grafo

- Ordenar as variáveis V_1, V_2, ..., V_n
- **Estado inicial**: uma associação vazia {} - nenhuma variável atribuída
- **Função sucessor**: atribuir um valor a uma variável desde que esta atribuição não entre em conflito com as anteriores
- **Teste de objetivo**: atribuição corrente é completa?
- **Custo do caminho**: constante para cada passo
- 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 para o problema do mapa

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



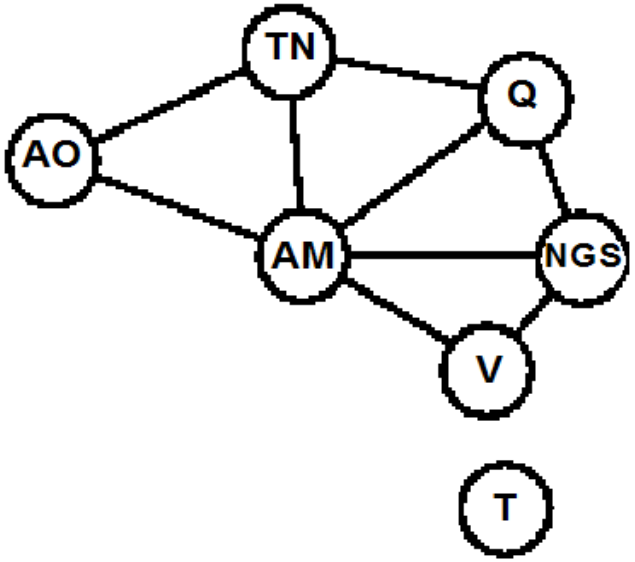
Exercício:
Continue a árvore de busca para gerar uma resposta para o problema do mapa da Austrália

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 também conhecida como 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**



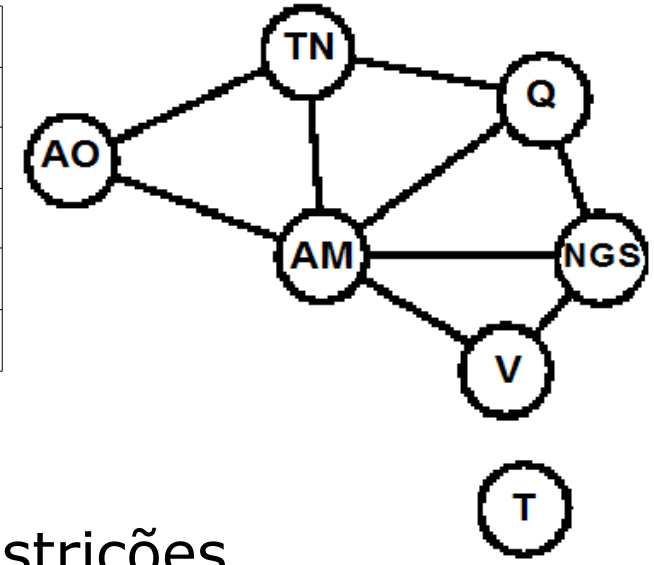
Heurísticas Gerais para PSRs

- 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

Utilizando verificação prévia e VRM

	AO	TN	Q	NGS	V	AM	T
DOMÍNIO	RGB	RGB	RGB	RGB	RGB	RGB	RGB
AO=R	**R**	GB	RGB	RGB	RGB	GB	RGB
Q=G	**R**	**B**	**G**	R B	RGB	**B**	RGB
V=B	**R**	**B**	**G**	**R**	**B**		RGB



- Verificação prévia não propaga todas as restrições
  - AO=R e Q=G, tanto TN quanto AM são forçados a serem B, mas TN e AM devem ter valores diferentes