

PCS 2428 / PCS 2059  
Inteligência Artificial

Prof. Dr. Jaime Simão Sichman  
Prof. Dra. Anna Helena Reali Costa

Satisfação de Restrições

Constraint Satisfaction Problems (CSP)

- Conceitos básicos
- Busca cega simples e refinada
- Busca heurística

2

Constraint Satisfaction Problems (CSP)

- Um Problema de Satisfação de Restrições
  - tipo de problema que impõe propriedades estruturais adicionais à solução a ser encontrada
  - há uma demanda mais refinada do que na busca clássica
    - ex. ir de Recife à Cajazeiras com no máximo 3 tanques de gasolina e 7 horas de viagem
- Um CSP consiste em:
  1. um conjunto de variáveis que podem assumir valores dentro de um dado domínio
  2. um conjunto de restrições que especificam propriedades da solução - valores que essas variáveis podem assumir.

3

Constraint Satisfaction Problems (CSP)

- Formulação
  - **Estados:** definidos pelos valores possíveis das variáveis
  - **Estado inicial:** nenhuma variável instanciada ainda
  - **Operadores:** atribuem valores (instanciação) às variáveis
    - uma variável por vez
  - **Teste de término:** verificar se todas as variáveis estão instanciadas obedecendo as restrições do problema
  - **Solução:** conjunto dos valores das variáveis instanciadas
  - **Custo de caminho:** número de passos de atribuição

4

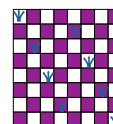
CSP: características das restrições

- O conjunto de valores que a variável  $i$  pode assumir é chamado domínio  $D_i$ 
  - O domínio pode ser **discreto** (fabricantes de uma peça do carro) ou **contínuo** (peso das peças do carro)
- Quanto à aridade, as restrições podem ser
  - unárias (sobre uma única variável)
  - binárias (sobre duas variáveis)
  - n-árias
    - a restrição unária é um sub-conjunto do domínio, enquanto que a n-ária é um produto cartesiano dos domínios
- Quanto à natureza, as restrições podem ser
  - **absolutas** (não podem ser violadas)
  - **preferenciais** (devem ser satisfeitas quando possível)

5

Exemplo

- Problema das 8-rainhas
  - *variáveis:* localização das rainhas (coluna, linha)
  - *valores:* possíveis posições do tabuleiro
  - *restrição binária:* duas rainhas não podem estar na mesma coluna, linha ou diagonal
  - *solução:* valores para os quais a restrição é satisfeita



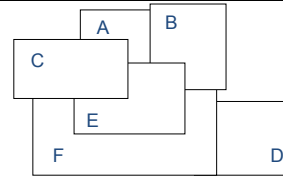
6

### Busca cega com Retrocesso para CSP

- Funcionamento
  - estado inicial: variáveis sem atribuição
  - aplica operador: instancia **uma** variável
  - teste de parada: todas variáveis instanciadas sem violações
- Retrocesso (*Backtracking*)
  - depois de realizar uma atribuição, verifica se restrições não são violadas
  - caso haja violação ⇒ retrocede
- Análise
  - pode ser **busca em profundidade limitada** ( $l =$  número de variáveis)
  - é completa
  - fator de expansão:  $\sum_i |D_i|$
  - o teste de parada é decomposto em um conjunto de restrições sobre as variáveis

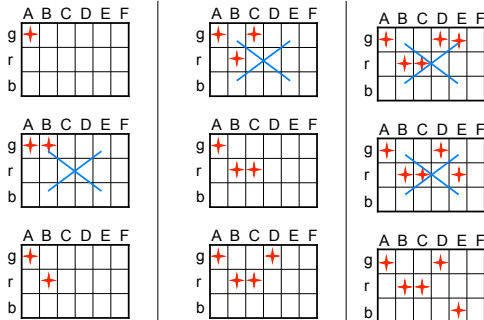
### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio:  $D_a=D_b\dots=D_f=\{\text{green,red,blue}\}$   
 restrições:  $A \neq B; A \neq C; A \neq E; B \neq E; B \neq F; C \neq E; C \neq F; D \neq F; E \neq F$

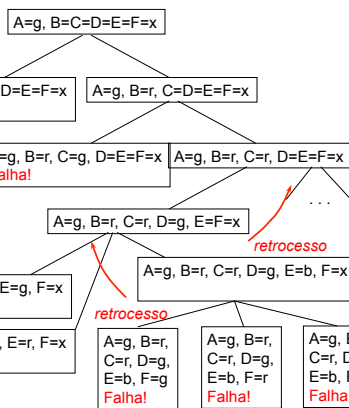
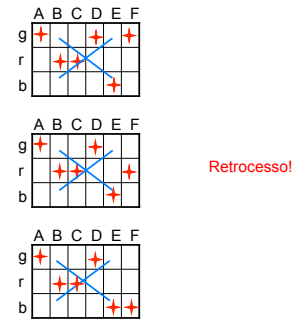


Solucionar usando busca em profundidade limitada com  $l=6$ .

### Simulação



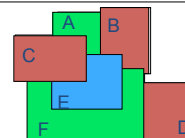
### Simulação



### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio:  $D_a=D_b\dots=D_f=\{\text{green,red,blue}\}$   
 restrições:  $A \neq B; A \neq C; A \neq E; B \neq E; B \neq F; C \neq E; C \neq F; D \neq F; E \neq F$

Simulando passo a passo:



1. A = green
2. B = green (falha c/ A)
3. B = red
4. C = green (falha c/ A)
5. C = red
6. D = green
7. E = green (falha c/ A)
8. E = red (falha c/ B)
9. E = blue
10. F = green (falha c/ D)
11. F = red (falha c/ C)
12. F = blue (falha c/ E) **BT**
13. E (falha) **BT**
14. D = red
15. E = green (falha c/ A)
16. E = red (falha c/ B)
17. E = blue
18. F = green

### Exemplo: coloração de mapas

Nova ordem, novo resultado →

variáveis: A,B,C,D,E,F  
 domínio: Da=Df={red,green,blue},  
 Db=Dd={green,blue,red},  
 Dc=De={blue, green, red}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

13 Solucionar usando busca em profundidade limitada com l=6.

### Exemplo: coloração de mapas

- A=red
- B=green
- C=blue
- D=green
- E= blue (falha com C)
- E= green (falha com B)
- E= red (falha com A) **BT**
- D=blue
- E= blue (falha com C)
- E= green (falha com B)
- E= red (falha com A) **BT**
- D=red
- E= blue (falha com C)
- E= green (falha com B)
- E= red (falha com A) **BT**
- D (falha) **BT**
- C= green
- D= green
- E= blue
- F=red

Mais Retrocessos

variáveis: A,B,C,D,E,F  
 domínio: Da=Df={red,green,blue},  
 Db=Dd={green,blue,red},  
 Dc=De={blue, green, red}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

### Backtracking não basta...

- Problema do *backtracking*:
  - não adianta mexer na 7a. rainha para poder posicionar a última
  - O problema é mais em cima... O *backtrack* normalmente tem que ser de mais de um passo
- Soluções:
  - Verificação prévia (*forward checking*)
  - Propagação de restrições

15

### Verificação Prévia

- Verificação prévia (*forward checking*)
  - idéia**: olhar para frente para detectar situações insolúveis
- Algoritmo:
  - Após cada atribuição, **elimina do domínio** das variáveis não instanciadas os valores incompatíveis com as atribuições feitas até agora.
  - Se um domínio torna-se vazio, retrocede imediatamente.
- É bem mais eficiente!

16

### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green, red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

17 Solucionar usando busca em profundidade limitada com l=6 e verificação prévia.

### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green,red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

Simulando passo a passo:

- A= green  
B,C,E={r,b}, D,F={g,r,b}
- B=red  
C={r,b}, D={g,r,b}, E={b}, F={g,b}
- C= red  
D={g,r,b}, E={b}, F={g,b}
- D=green  
E={b}, F={b}
- E= blue  
F=?? **BT**
- E=?? **BT**
- D=red  
E={b}, F={g,b}
- E= blue  
F={g}
- F=green

18

### Propagação de restrições

- Propagação de restrições (constraint propagation)
  - uma consequência da verificação prévia
  - quando um valor é eliminado, isto é **propagado** para outros valores que dele dependem, podendo torná-los inconsistentes e eliminados também
  - é como uma onda que se propaga: as escolhas ficam cada vez mais restritas

19

### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green, red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

Solucionar usando busca em profundidade limitada com l=6 e verificação prévia combinada com propagação de restrições

20

### Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green, red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

Simulando passo a passo:  
 1. A=green  
 B,C,E={r,b}, D,F={g,r,b}  
 2. B=red  
 C={r,b}, D={g,r,b}, E={b}, F={g,b}  
 → C={r}, F={g} → D={r,b}  
 3. C=red  
 D={r,b}, E={b}, F={g}  
 4. D=red  
 E={b}, F={g}  
 5. E=blue  
 6. F=green  
 Sem retrocesso!

21

### Heurísticas para CSP

- Tenta reduzir o fator de expansão do espaço de estados
- Onde pode entrar uma heurística?
  - Ordenando a escolha da **variável** a instanciar
  - Ordenando a escolha do **valor** a ser associado a uma variável
- Existem 3 heurísticas para isto...
  - Variável mais restritiva**: variável envolvida no maior número de restrições é preferida
  - Variável mais restringida**: variável que pode assumir menos valores é preferida
  - Valor menos restritivo**: valor que deixa mais liberdade para futuras escolhas

22

### Variável mais restritiva

(variável envolvida no maior número de restrições)

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green, red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

Candidatas!: E, F, ...resto  
 E = green  
 Candidatas: F, ...resto  
 F = red  
 Candidatos: A, B, C, D  
 A= red  
 Candidatos: B, C, D  
 B= blue  
 Candidatos: C, D  
 C= blue  
 D= green  
 SEM BACKTRACK!!

23 1 em ordem de prioridade (4, 4, 3, 3, 3, 1)

### Coloração de mapas: variável mais restringida

(variável que pode assumir menos valores)

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green, red, blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

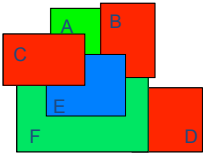
Candidatas: todas  
 A = green  
 Candidatas: B(rb), C(rb), E(rb), ...  
 B = red  
 Candidatos: E(b), C(rb), F(gb), ...  
 E=blue  
 Candidatos: C(r), F(g), D...  
 C=red  
 Candidatos: F(g), D(rgb)  
 F=green  
 D = blue ou red  
 SEM BACKTRACK!!

24

Coloração de mapas: valor menos restritivo  
(valor que deixa mais liberdade)

1. A = green B(br), C(br), D(grb), E(br), F(grb)  
 2. B=r ou B=b "poda" liberdade igualmente → faz escolha na ordem  
 B = red C(rb), D(grb), E(b), F(gb)  
 3. C=r: D(grb), E(b), F(gb); C=b: D(grb), E( ), F(g) → escolhe C=r  
 C = red D(grb), E(b), F(gb)  
 4. D=g: E(b), F(b); D=r: E(b), F(gb); D=b: E(b), F(g) → escolhe D=r  
 D = red E(b), F(gb)  
 5. E = blue F(g)  
 6. F=green  
 SEM BACKTRACK!!

variáveis: A,B,C,D,E,F  
 domínio: Da=Db...=Df={green,red,blue}  
 restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F



## CSP – conclusões

- Grande importância prática, sobretudo em tarefas de
  - Criação, projeto (design)
  - Agendamento (scheduling)
  - onde várias soluções existem e é mais fácil dizer o que não se quer...
- Estado atual
  - Grandes aplicações industriais \$\$\$\$
  - Número crescente de artigos nas principais conferências
- Observação:
  - a sigla CSP também é usada para falar de **Constraint Satisfaction Programming**, que é um paradigma de programação

26