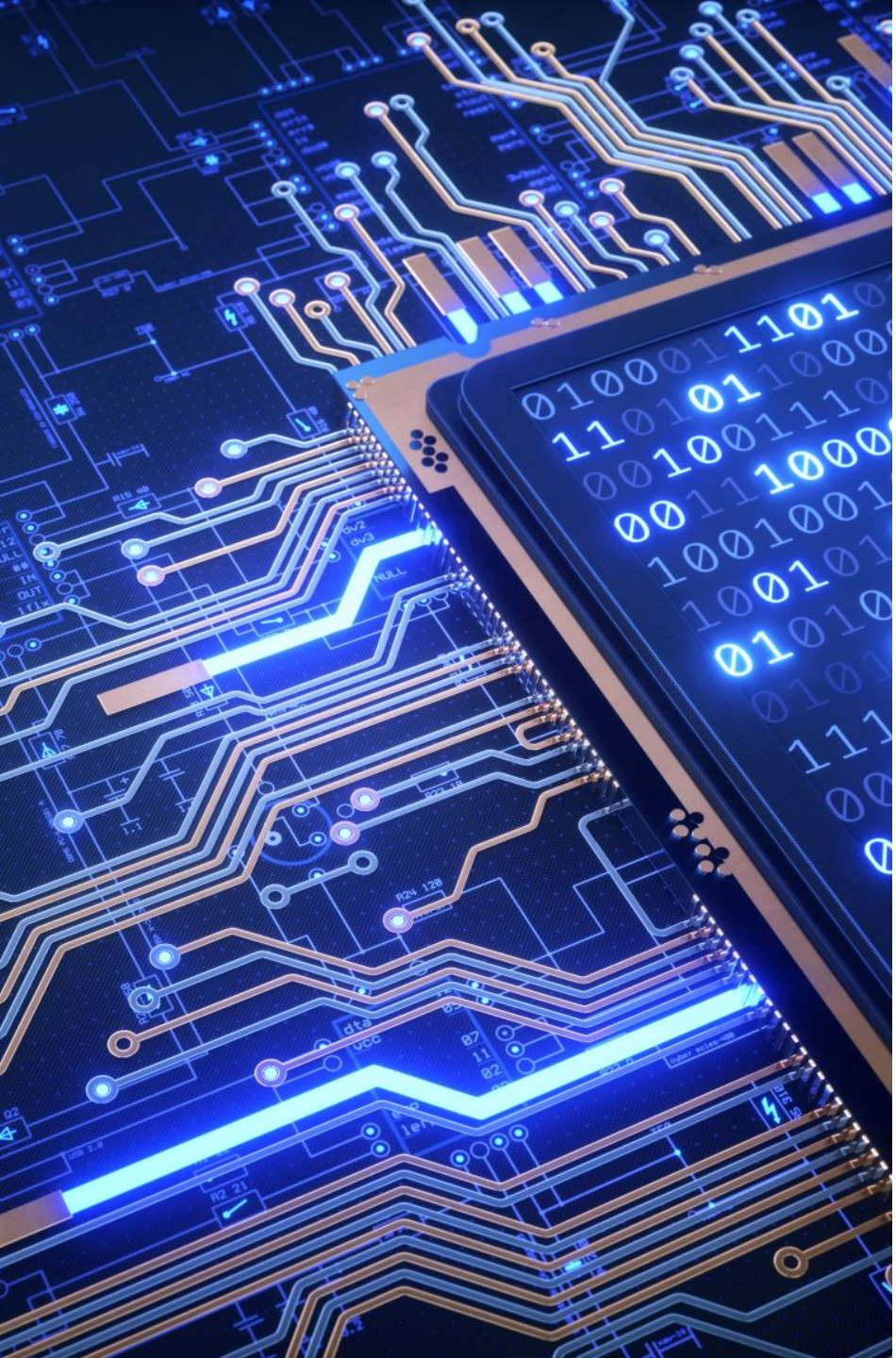


Problemas de Satisfação de Restrições

Inteligência Artificial

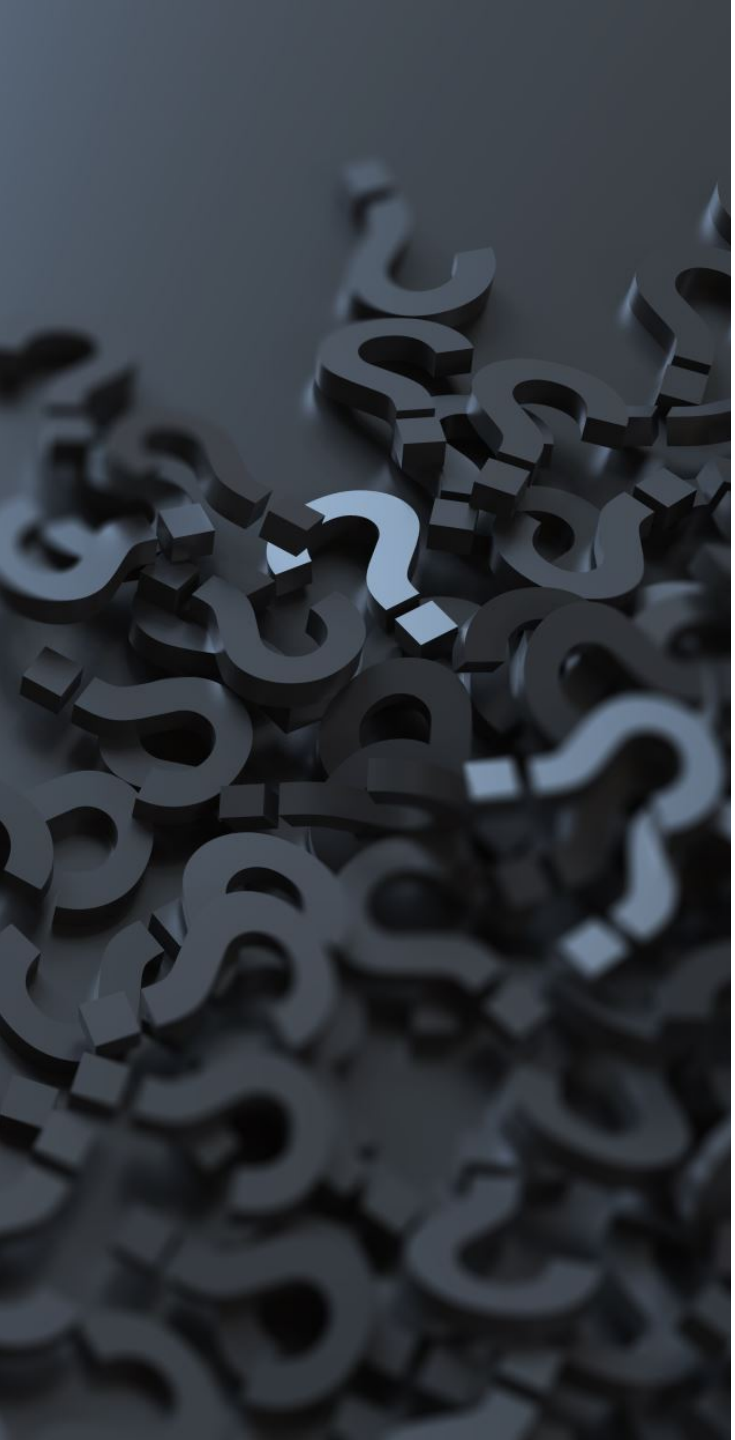
PCS3438

Constraint Satisfaction Problems (CSP)

Problemas de Satisfação de Restrições

Plano de aula:

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



CSP

- Um Problema de Satisfação de Restrições
  - impõe **propriedades estruturais** adicionais à solução a ser encontrada
  - há uma demanda mais refinada do que na busca clássica
    - ex. ir de São Paulo à Santa Fé do Sul com no máximo 3 tanques de gasolina e 7 horas de viagem

CSP

- Um CSP 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 as variáveis podem assumir.

CSP: formulação

- **Estados:** definidos pelos ***valores*** possíveis das variáveis
- **Estado inicial:** nenhuma variável instanciada
- **Operadores:** atribuem valores (instanciação) às variáveis
  - **Uma variável por vez**
- **Teste de término:** quando 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

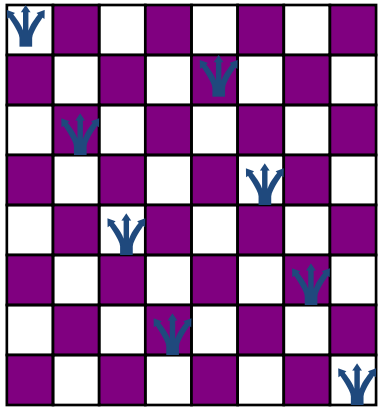
CSP: características das restrições

- O domínio *Di* da variável *i*:
  - 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 subconjunto 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)

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

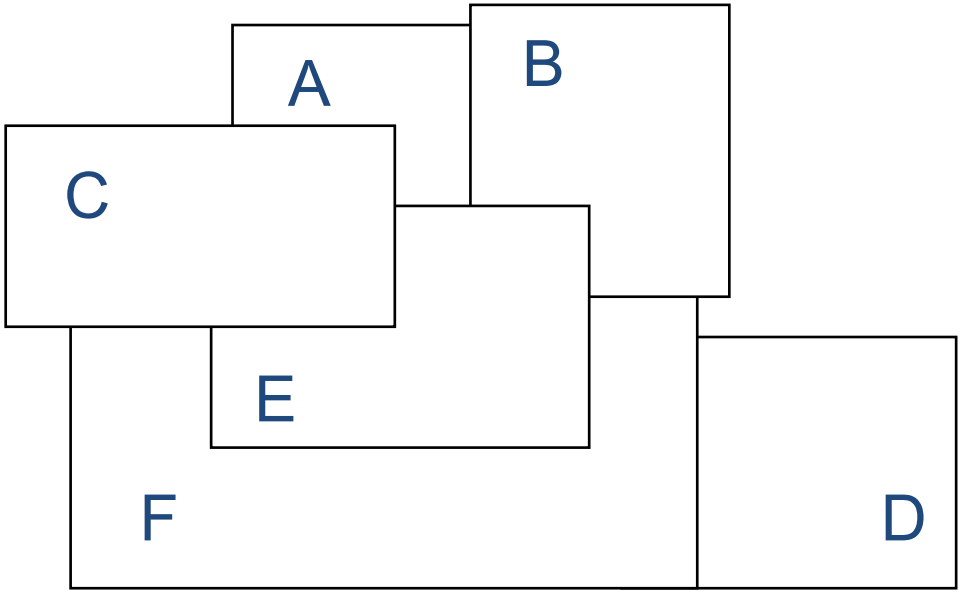


Busca Cega com Retrocesso para CSP

- **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 $\Rightarrow$ **retrocede**
- **Análise**
  - pode ser **busca em profundidade limitada** (l = número de variáveis)
  - é completa
  - fator de ramificação máxima: max(|Di|)
  - o teste de parada é decomposto em um conjunto de restrições sobre as variáveis

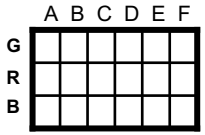
Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={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.

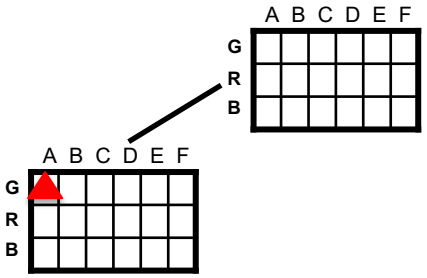
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

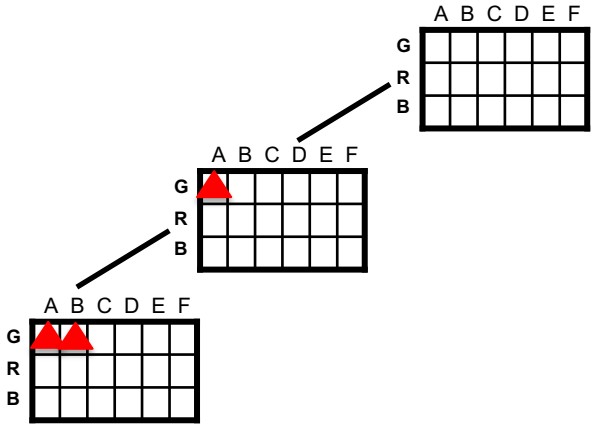
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

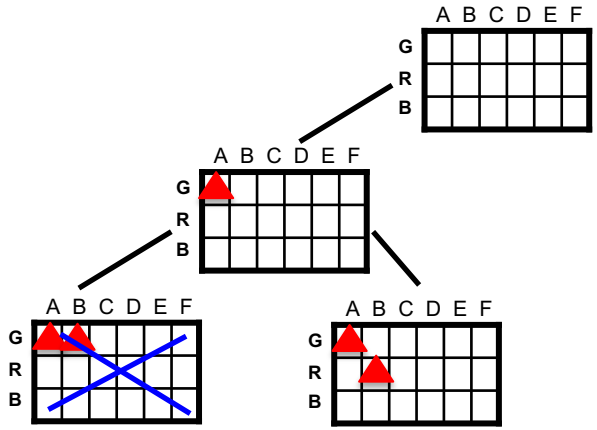
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

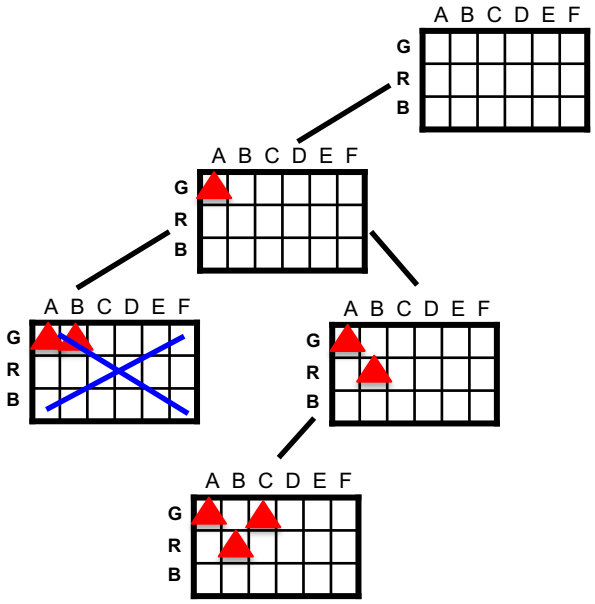
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

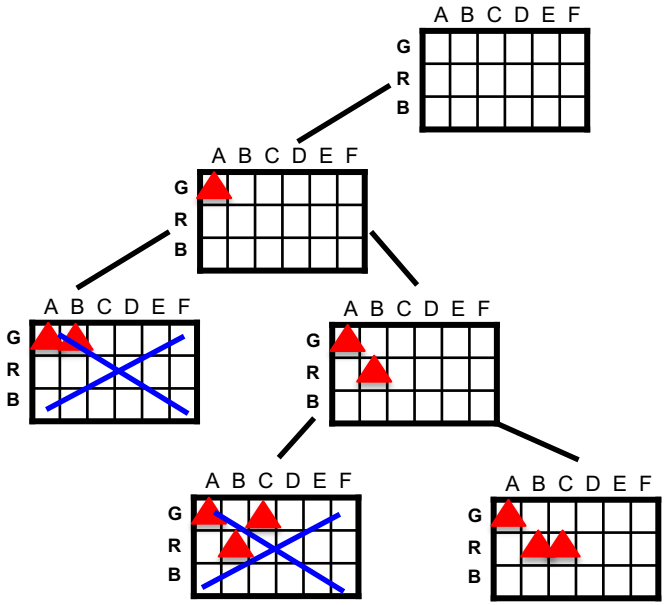
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

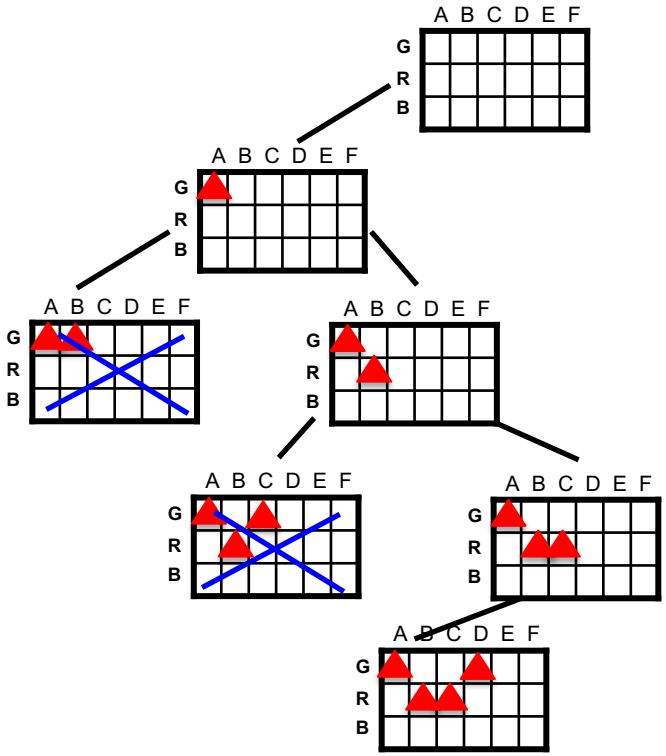
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

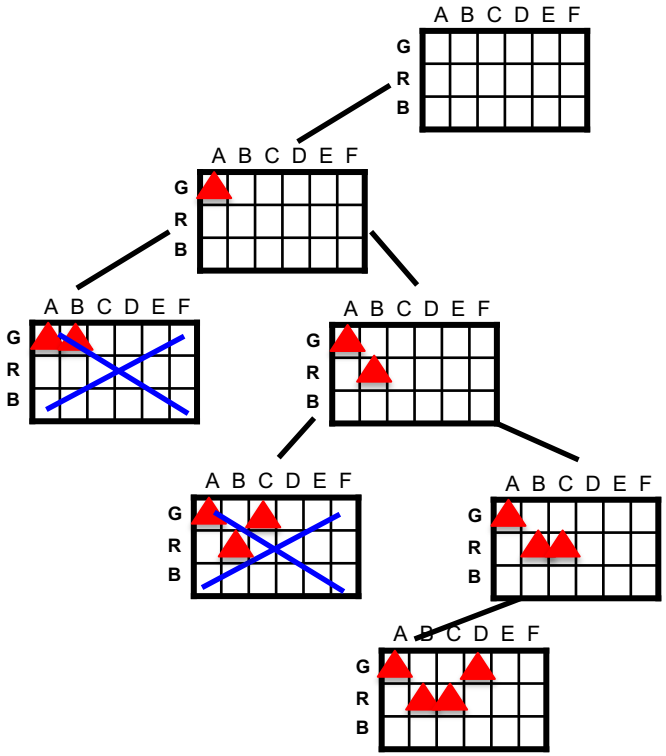
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

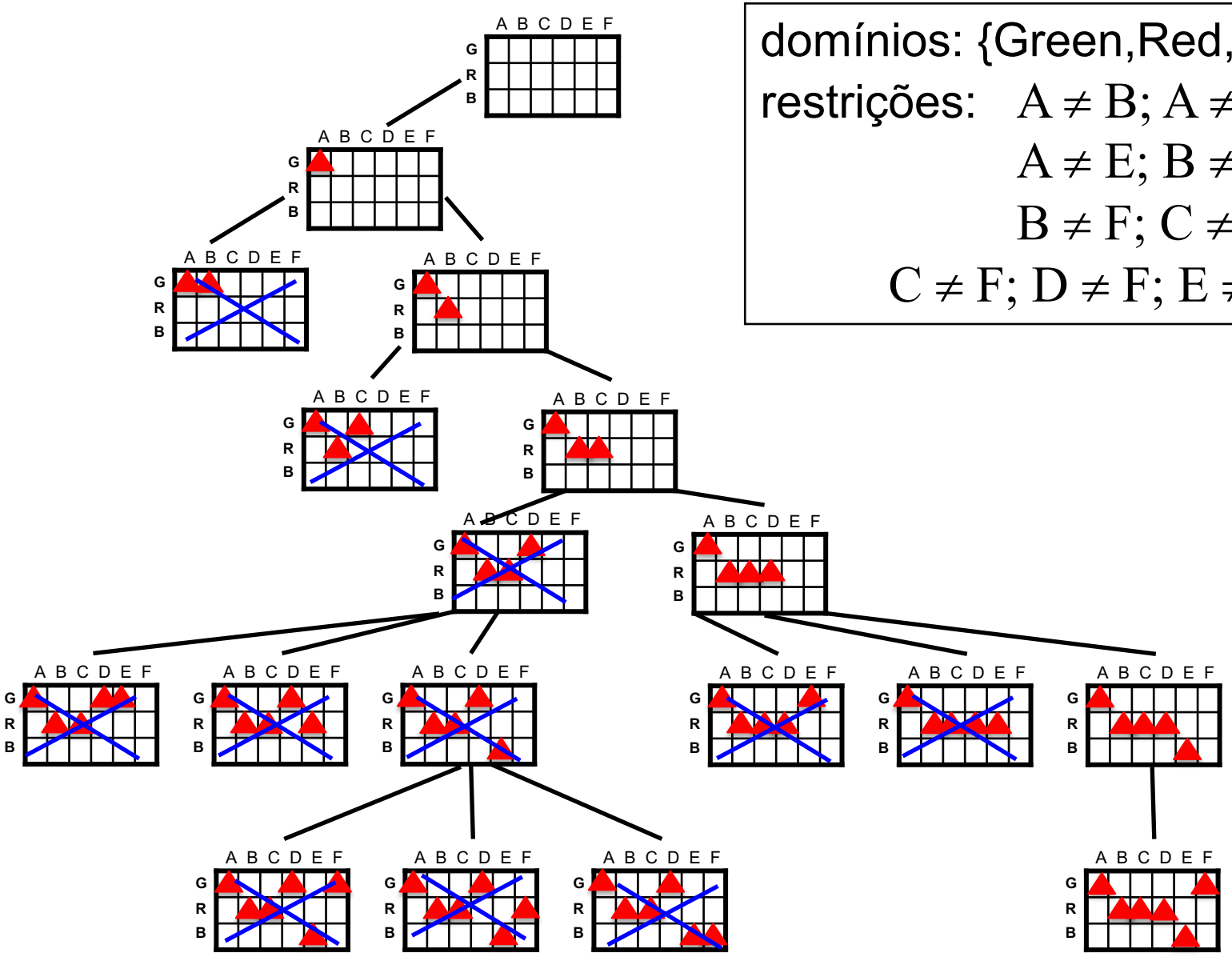
Árvore de Busca



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

Árvore de Busca



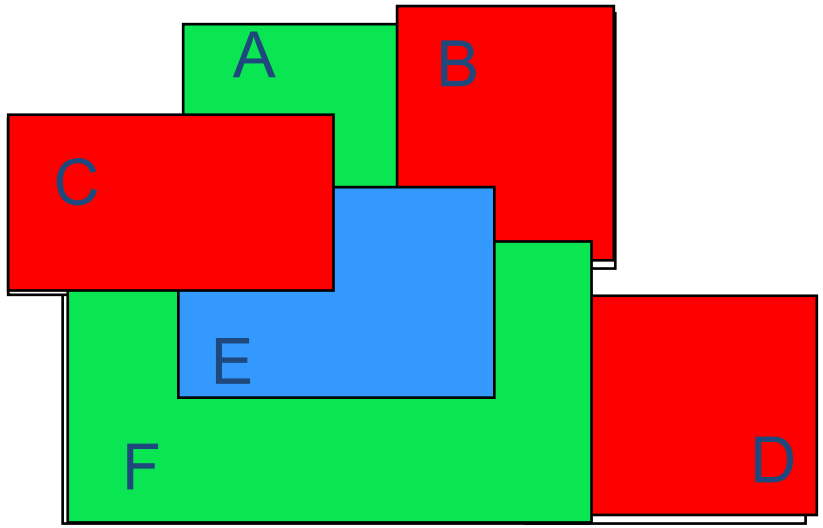
domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

Exemplo: coloração de mapas

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

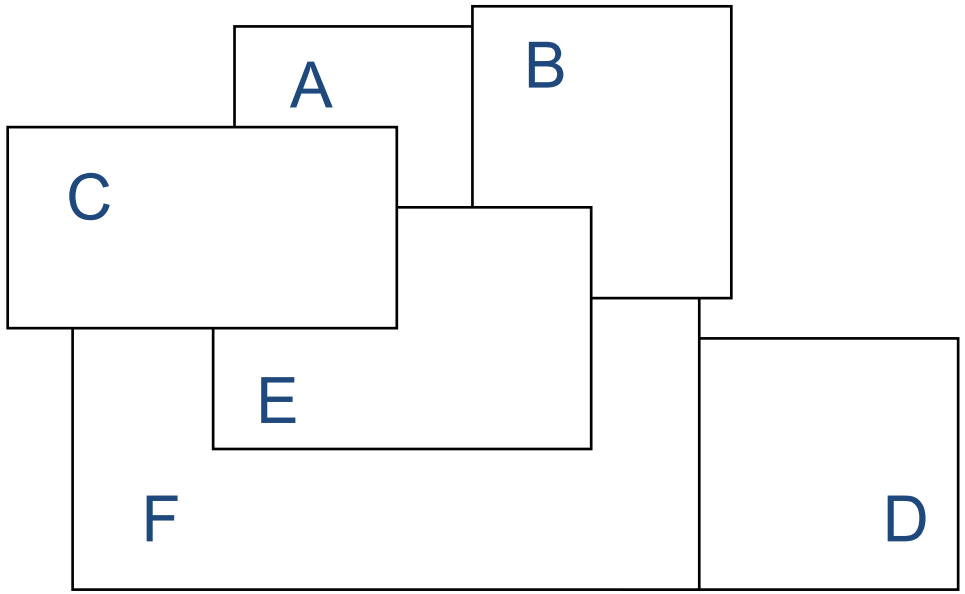
restrições: $A \neq B$; $A \neq C$; $A \neq E$;
$B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$;
$D \neq F$; $E \neq F$



Exemplo: coloração de mapas

Nova ordem, novo resultado

variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={Red, Green, Blue}
restrições: $A \neq B$; $A \neq C$; $A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$; $C \neq F$; $D \neq F$; $E \neq F$



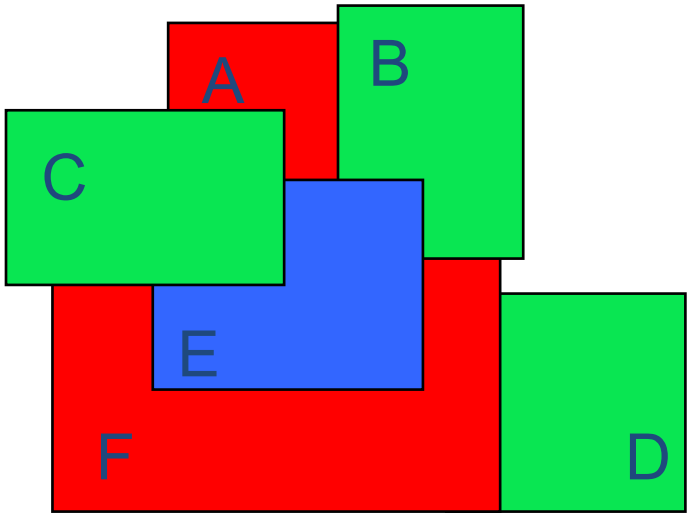
Solucionar usando busca em profundidade limitada com l=6.

Exemplo: coloração de mapas

Mas poderia começar por red e fazer outra forma de seleção de valores no domínio ...

A=red
B=green
C=blue
D=red
E= ?? Backtracking
D=green
E=?? Backtracking
D=blue
E=?? Backtracking
D= ?? Backtracking
C = green
D = green
E = blue
F=red

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



Backtracking não basta...

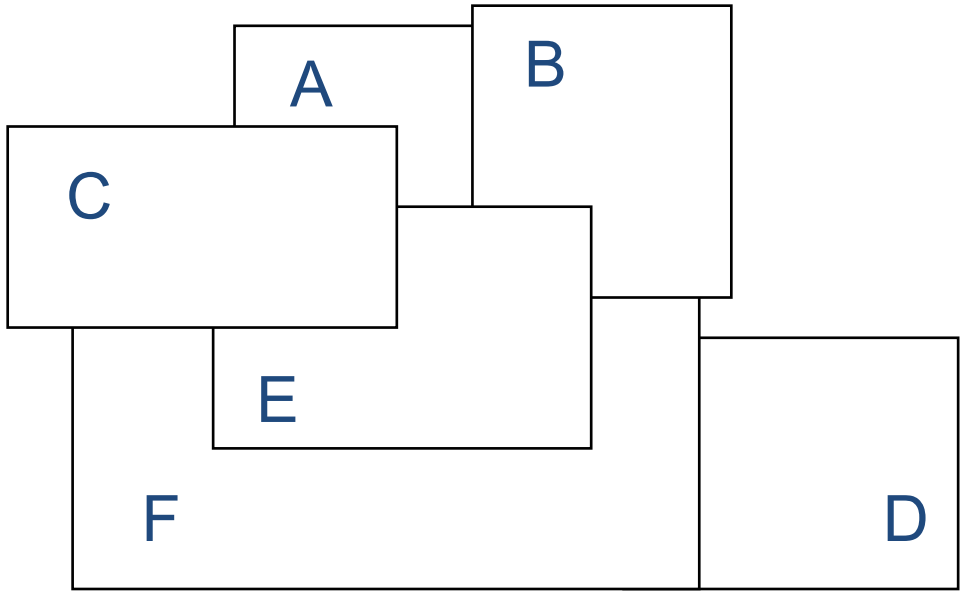
- Problema do *backtracking*:
  - não adianta mexer na 7a. rainha para poder posicionar a última.
  - O problema é mais em cima... O retrocesso normalmente tem que ser de mais de um passo
- Soluções:
  - **Verificação prévia *(forward checking)***
  - **Propagação de restrições**

Verificação Prévia

- Verificação prévia (*forward checking*)
  - **idéia**: olhar para frente para detectar situações insolúveis
- Algoritmo:
  - Após cada atribuição, **eliminar 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!

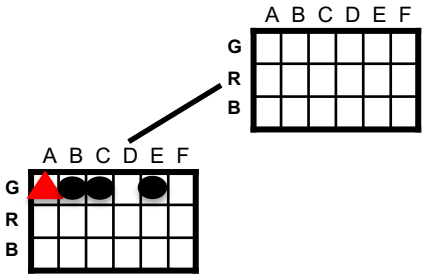
Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={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 e **verificação prévia**.

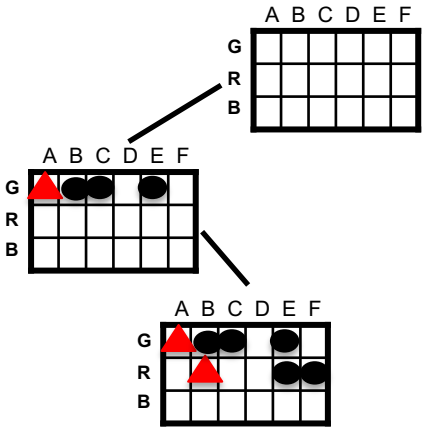
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

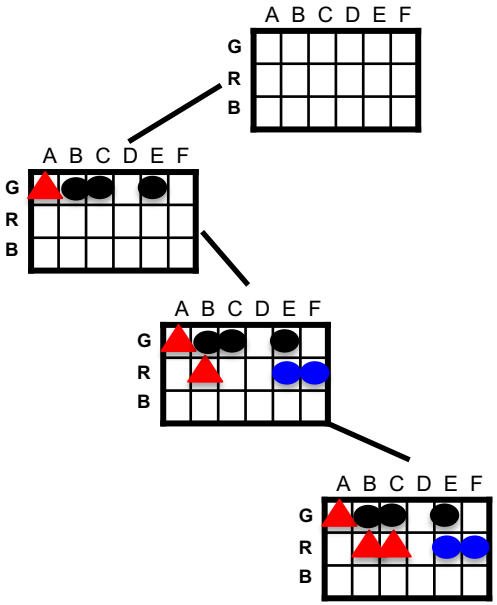
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

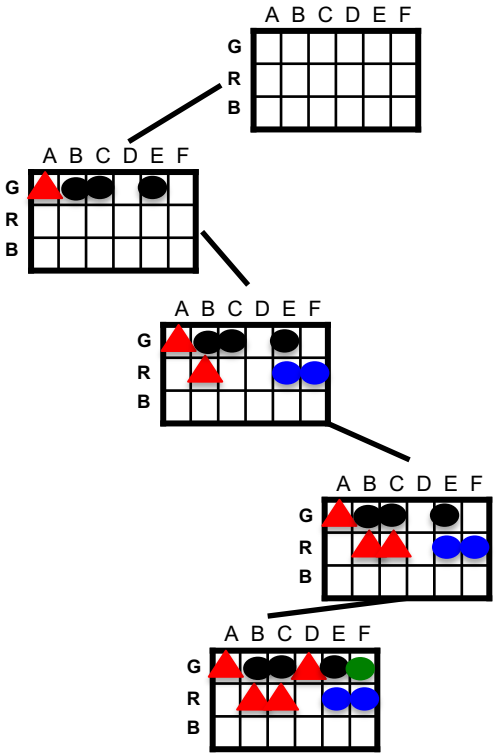
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

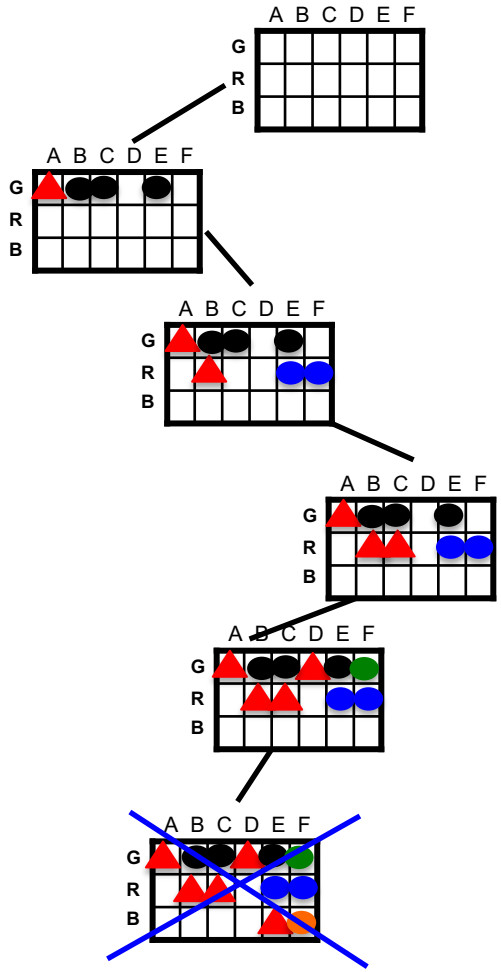
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

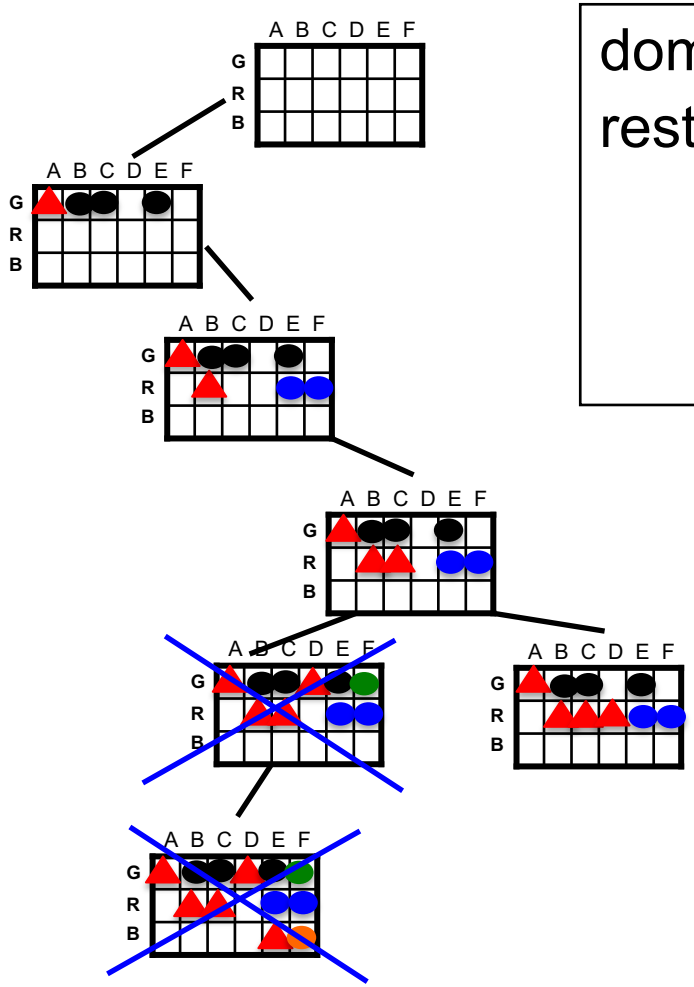
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

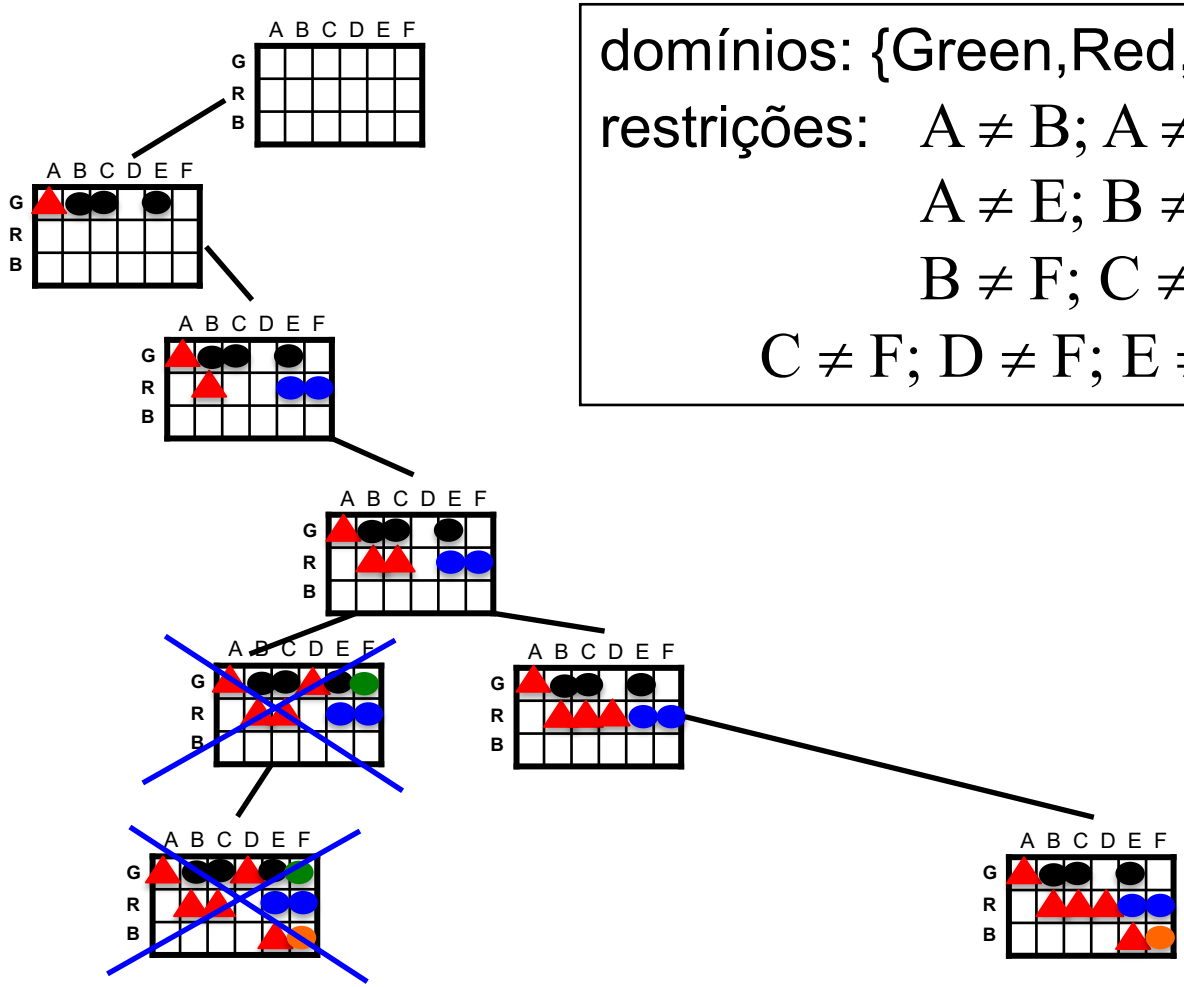
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

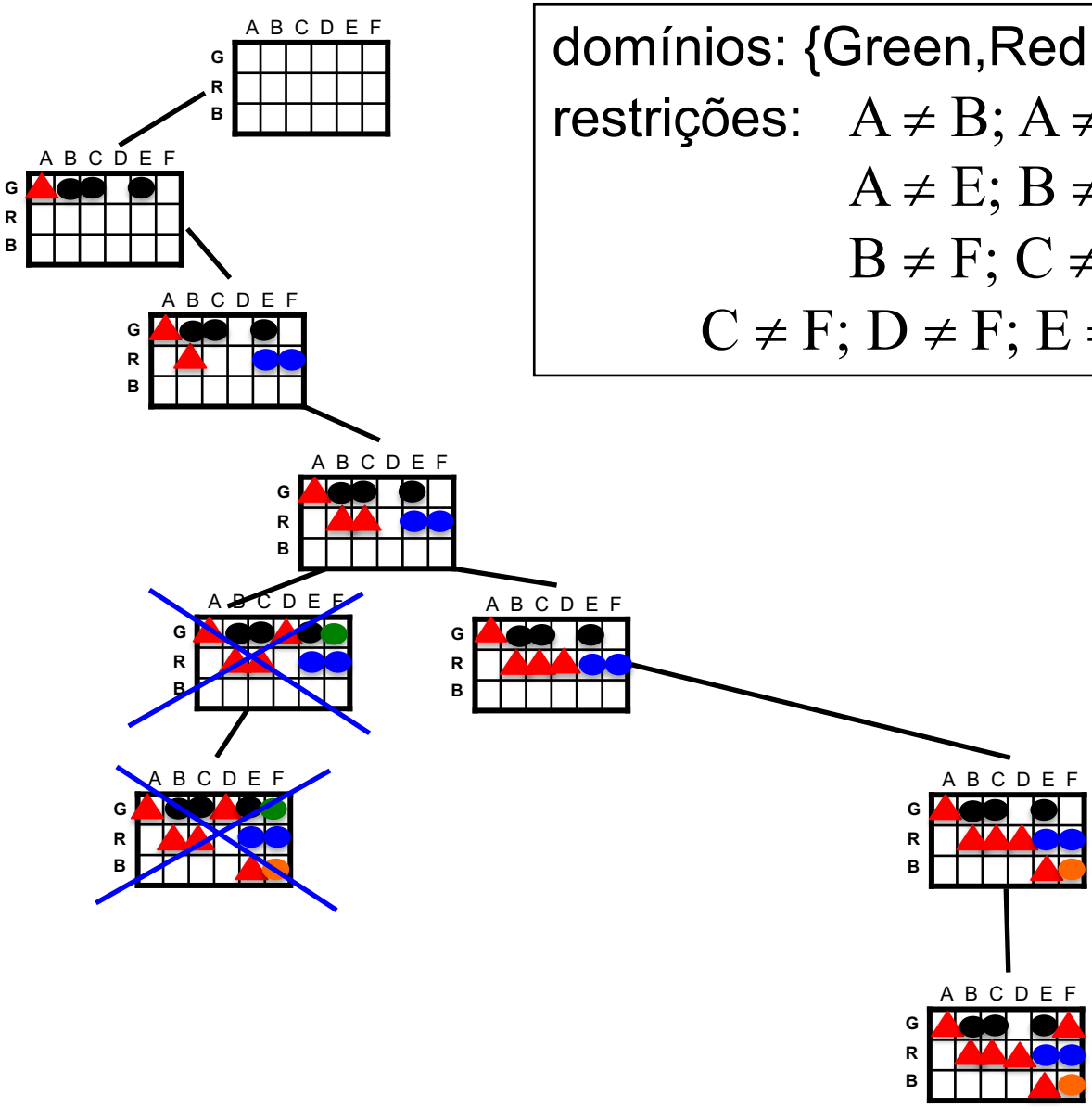
Árvore de Busca (com V.P.)



domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

Árvore de Busca (com V.P.)



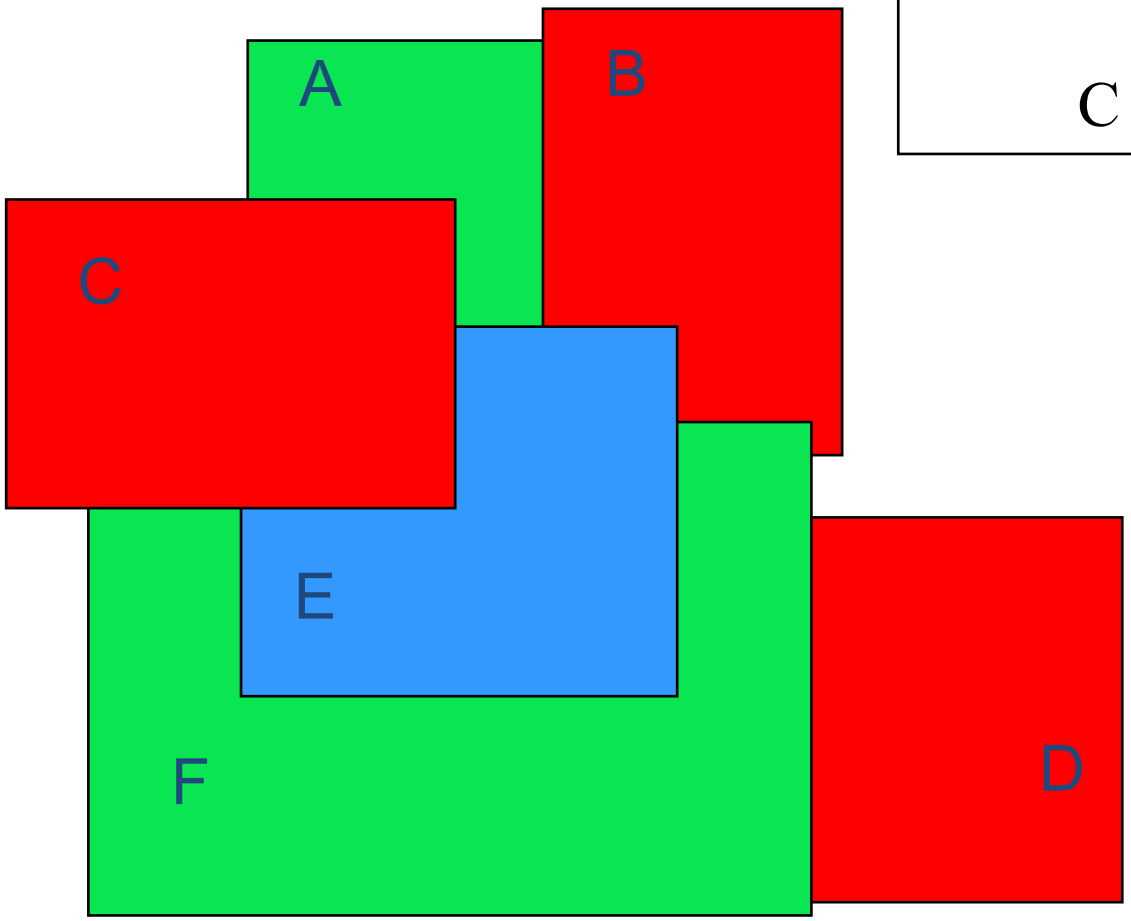
domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

CSP com verificação prévia

domínios: {Green,Red,Blue}

restrições: $A \neq B$; $A \neq C$;
$A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$;
$C \neq F$; $D \neq F$; $E \neq F$

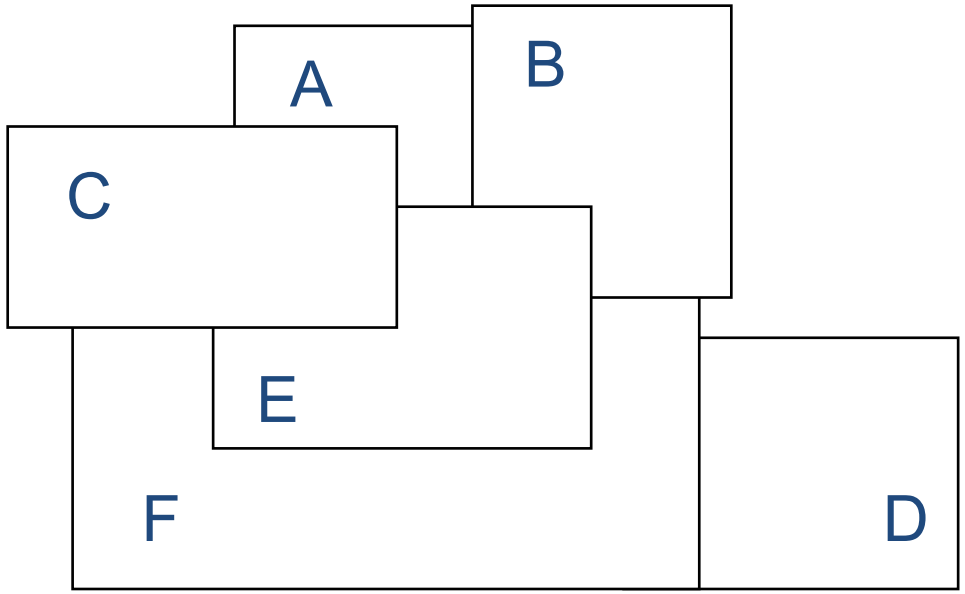


Propagação de restrições

- 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

Exemplo: coloração de mapas

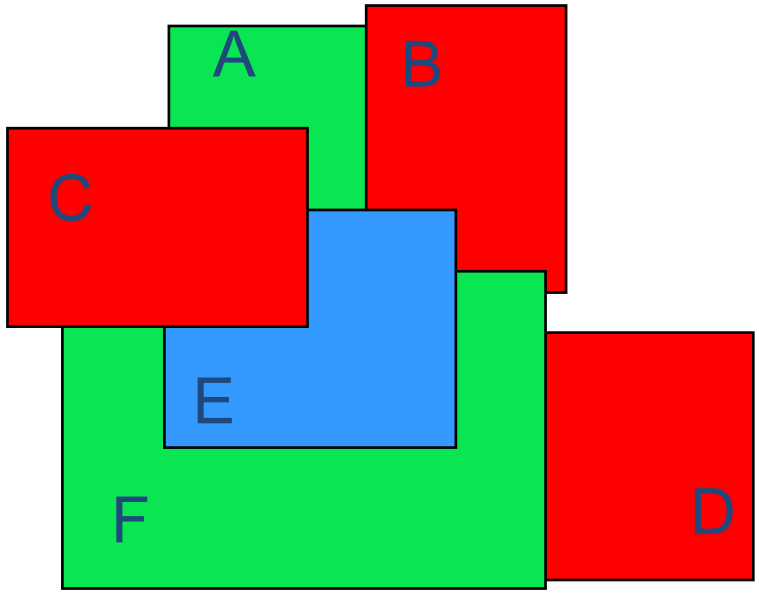
variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={Green, Red, Blue}
restrições: $A \neq B$; $A \neq C$; $A \neq E$; $B \neq E$;
$B \neq F$; $C \neq E$; $C \neq F$; $D \neq F$; $E \neq F$



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

Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F
domínio:
Da=Db...=Df={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 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!

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

Heurísticas para CSP

- Existem 3 heurísticas para isto...
  - ***Variável mais restritiva***: variável envolvida no maior número de restrições é preferida (verifica restrições)
  - ***Variável mais restringida***: variável que pode assumir menos valores é preferida (verifica os domínios das variáveis)
  - ***Valor menos restritivo***: valor que deixa mais liberdade para futuras escolhas (verifica os valores dos domínios e as restrições)

Variável mais restritiva

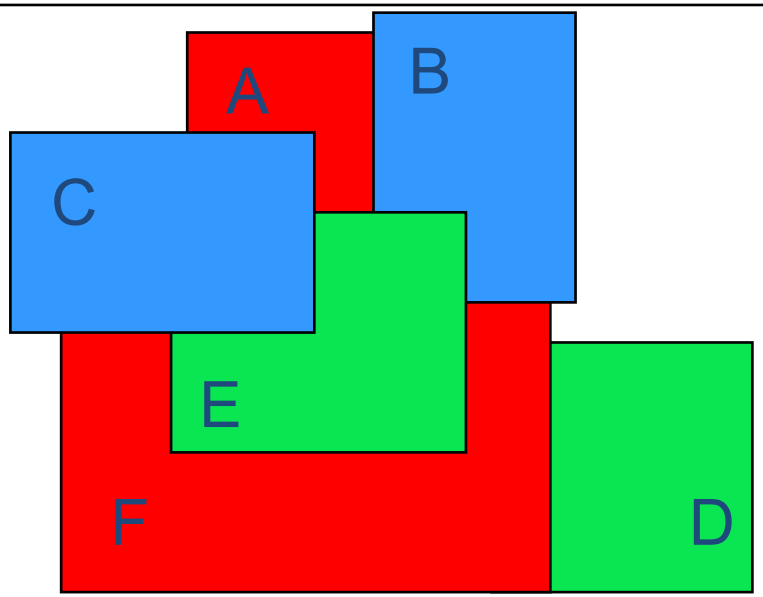
(variável envolvida no maior número de restrições)

Simulação com verificação prévia e heurística:

Candidatas: E, F, ...resto
E = green A=B=C=F={r,b}, D={g,r,b}
Candidatas: F, ...resto
F = red A={r,b} B=C={b}, D={g,b}
Candidatas: A, B, C, D
A= red
Candidatas: B, C, D
B= blue
Candidatas: C, D
C= blue
D = green

SEM BACKTRACK!!

variáveis: A,B,C,D,E,F
Domínio:
Da=Db...=Df={green,red,blue}
Restrições: $A \neq B$; $A \neq C$; $A \neq E$;
$B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$;
$D \neq F$; $E \neq F$



Candidatas em ordem de prioridade (E=4, F=4, A=3, B=3, C=3, D=1)

Variável mais restringida

(variável que pode assumir menos valores)

Simulação com verificação prévia e heurística:

Candidatas: todas

A = green

Candidatas: B(rb), C(rb), E(rb), ...

B = red

Candidatas: E(b), C(rb), F(gb), ...

E = blue

Candidatas: C(r), F(g), D...

C = red

Candidatas: F(g), D(rgb)

F = green

Candidata: D(rb)

D = red

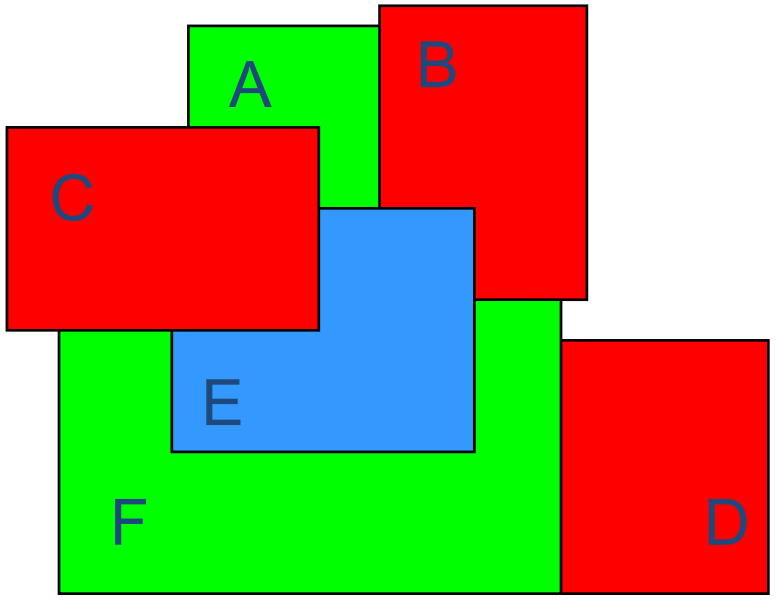
SEM BACKTRACK!!

variáveis: A,B,C,D,E,F

Domínio:

Da=Db...=Df={green,red,blue}

Restrições: $A \neq B$; $A \neq C$; $A \neq E$;
$B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$;
$D \neq F$; $E \neq F$



Valor menos restritivo

(valor que deixa mais liberdade)

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

1. A = green B(rb), C(rb), D(grb), E(rb), 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

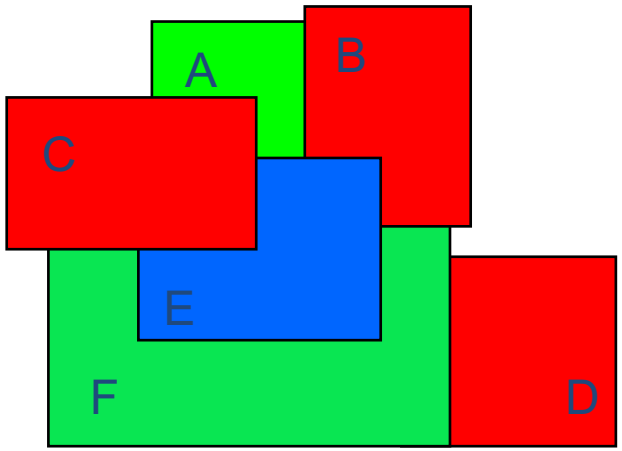
C= red D(grb), E(b), F(gb)

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!!



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 Programmimg***, que é um paradigma de programação

Bibliografia

- Capítulo 6 do livro texto
  - Russel & Norvig, Inteligência Artificial, 3a. Edição