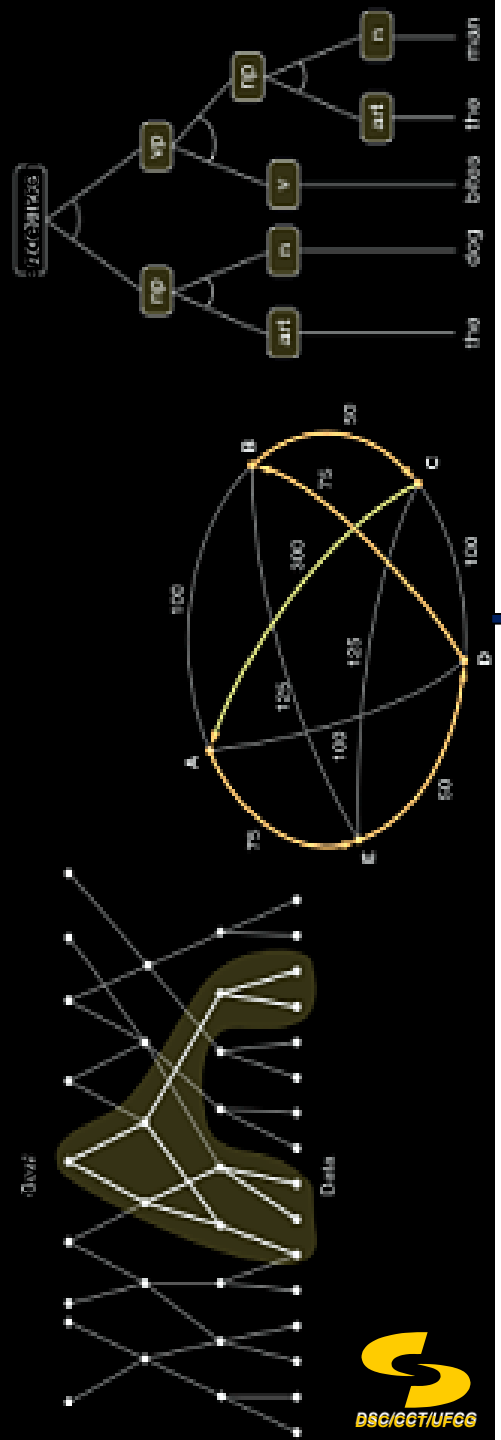
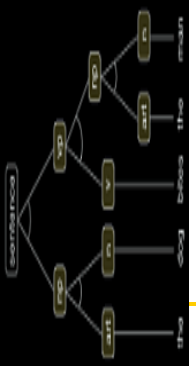


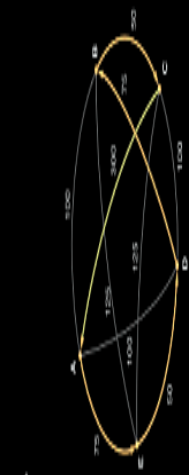
Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Pós-Graduação em Ciência da Computação

Inteligência Artificial

Resolução de Problemas (Parte V)

Prof.ª Joseana Macêdo Fechine
joseana@dsc.ufcg.edu.br



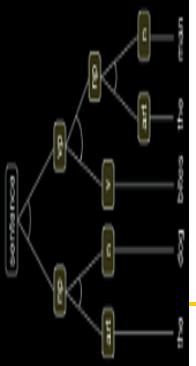
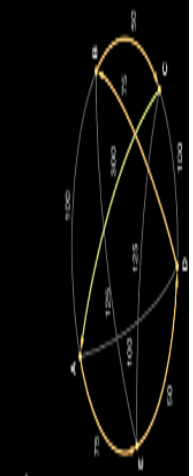


Em Busca de Soluções

Tópico

- **Problemas de Satisfação de Restrições**

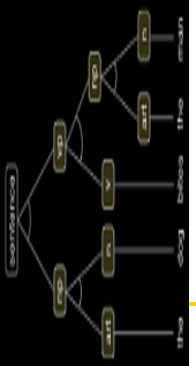
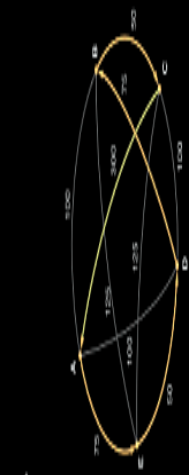
DSC/CCT/UFCG


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.
- Tem-se um conjunto de variáveis que podem assumir valores dentro de um dado domínio.
- Tem-se um conjunto de restrições que especificam propriedades da solução - valores que essas variáveis *podem* assumir.

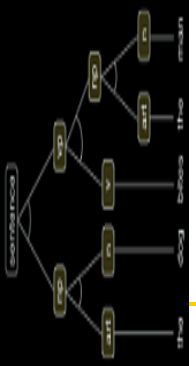
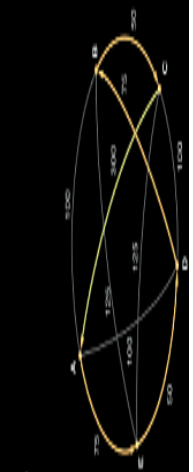

Constraint Satisfaction Problems (CSP)

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 às restrições do problema.
- **Solução:** conjunto dos valores das variáveis instanciadas.
- **Custo de caminho:** número de passos de atribuição.

Constraint Satisfaction Problems (CSP)

- O conjunto de valores que a variável ***i*** pode assumir é chamado **domínio *Di***
  - 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
- **Quanto à natureza, as restrições podem ser**
  - **absolutas** (não podem ser violadas)
  - **preferenciais** (devem ser satisfeitas quando possível)

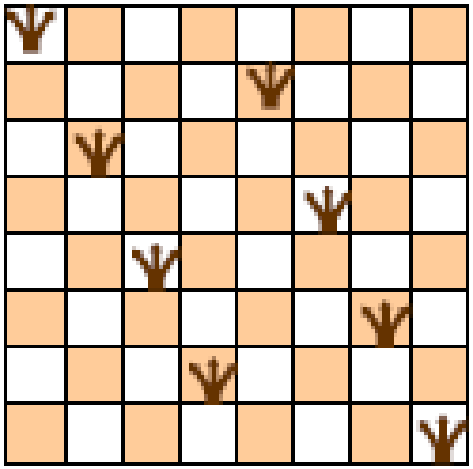

Constraint Satisfaction Problems (CSP)

- Toda solução deve ser uma atribuição completa e, portanto, aparece na profundidade *n*.
- A árvore de busca se estende até a profundidade *n.*
- Por essas razões, os algoritmos de busca em profundidade são populares para CSP.
- **O caminho pelo qual uma solução é alcançada é irrelevante**.

Constraint Satisfaction Problems (CSP)

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



Constraint Satisfaction Problems (CSP)

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

Constraint Satisfaction Problems (CSP)

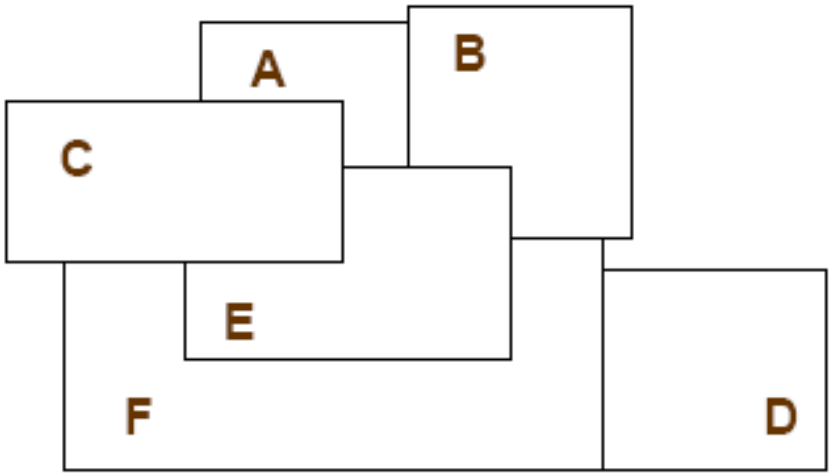
Busca cega com Retrocesso para CSP

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

Constraint Satisfaction Problems (CSP)

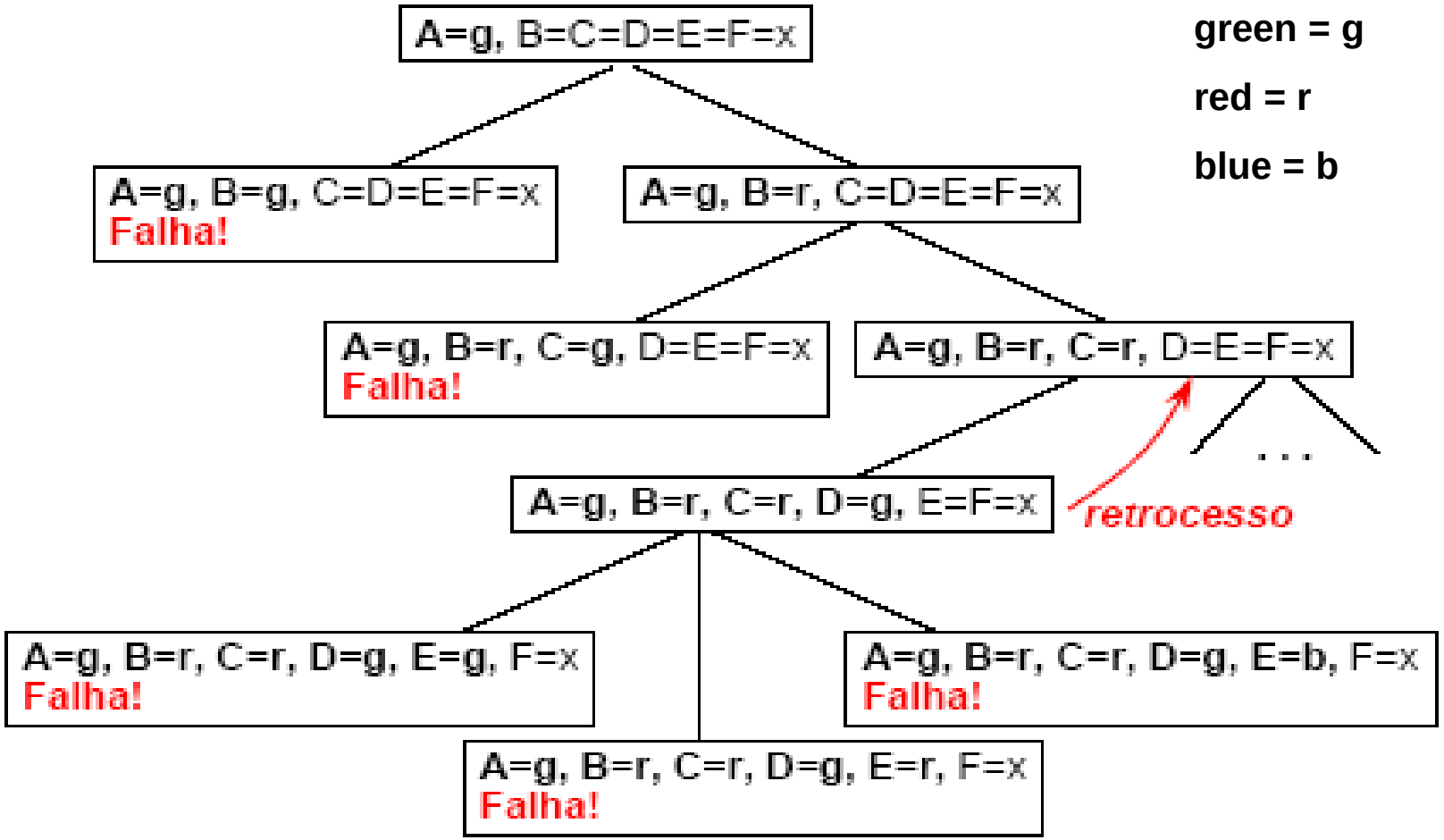
Exemplo: coloração de mapas

- **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.

Constraint Satisfaction Problems (CSP)

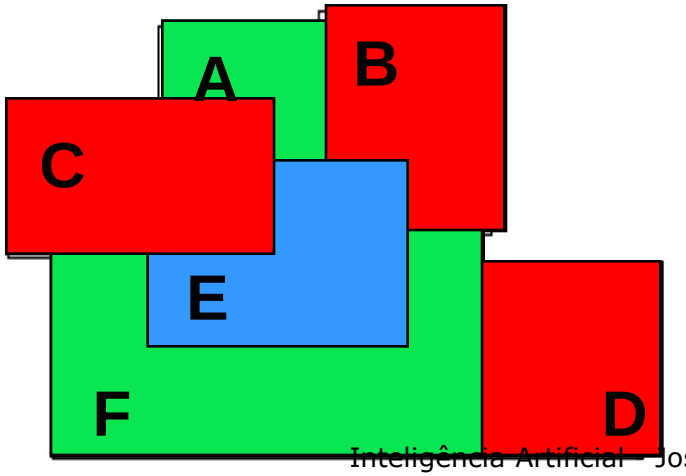


Solucionar usando busca em profundidade limitada com l=6 e verificação prévia combinada com propagação de restrições

Constraint Satisfaction Problems (CSP)

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



Simulação passo a passo...

A= green

B = green (falha c/ A)

B=red

C=green (falha c/ A)

C= red

D=green

E= green (falha c/ A)

E= red (falha c/ B e C)

E= blue

F=green (falha c/ D)

F=red (falha c/ C)

F = blue (falha c/ E)

F backtracking

E backtracking

D=red

E=green (falha c/ A)

E= red (falha c/ B)

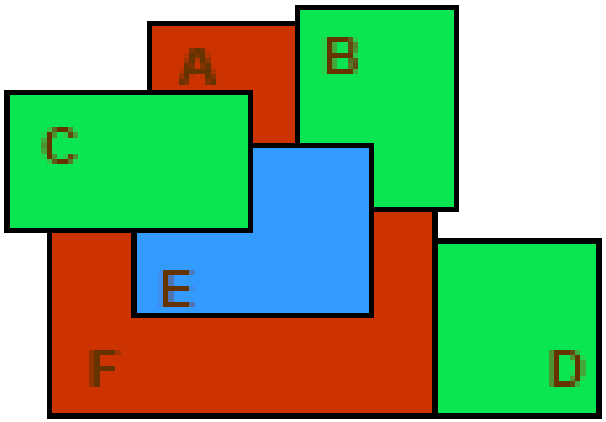
E= blue

F=green

Constraint Satisfaction Problems (CSP)

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

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



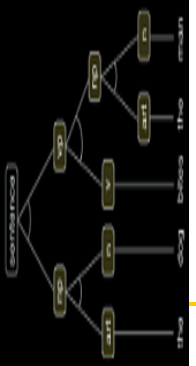
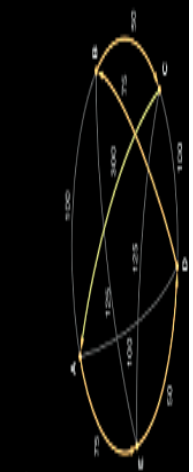
A=red
B=green
C=blue
D=red
E= ?? Backtracking
D=green
E=?? Backtracking
D=blue
E=?? Backtracking
D= ?? Backtracking
C = green
D = green
E = blue
F=red

Constraint Satisfaction Problems (CSP)

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

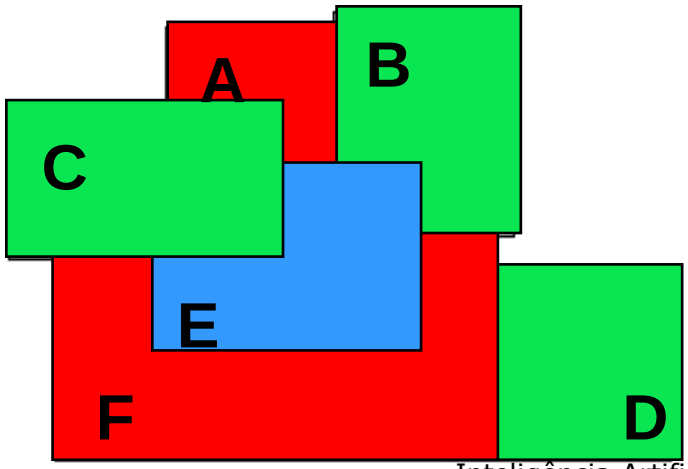

Constraint Satisfaction Problems (CSP)

- **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!**

Constraint Satisfaction Problems (CSP)

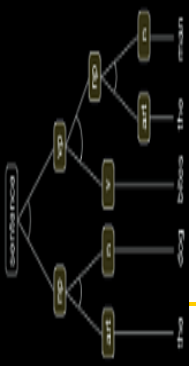
Exemplo: coloração de mapas

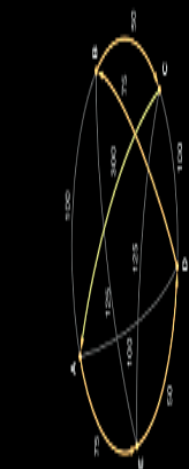
- **variáveis:** A,B,C,D,E,F
- **domínio:** Da=Db...=Df={red,green,blue}
- **restrições**: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F



Passo a passo...
A=red
=> B, C, E ={green,blue}
(variáveis c/ restrições c/ A)
=> D, F ={red,green,blue}
B=green
=> E = {blue}, F = {red, blue}
(variáveis c/ restrições c/ B)
=> C ={green,blue},
D ={red,green,blue}
C = green
=> E ={blue}, F = {red, blue}
(restrições c/ C)
=> D = {red,green,blue}
D=red, E=blue, F=??

Backtracking F e D!!
D=green, E=blue, F=red





Constraint Satisfaction Problems (CSP)

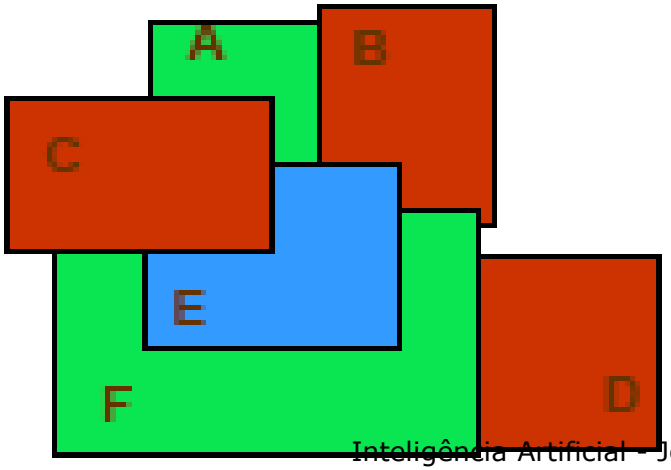
Propagação de restrições (*constraint propagation*)

- Uma conseqüência 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.

Constraint Satisfaction Problems (CSP)

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={r}, F={g} ⇒ D={r,b}
C= red D={r,b}, E={b}, F={g}
D=red E={b}, F={g}
E= blue
F=green
Sem retrocesso!

Constraint Satisfaction Problems (CSP)

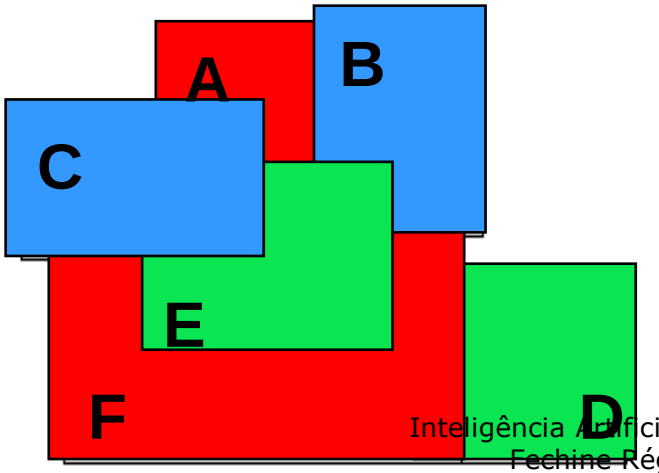
Heurística para CSP

- **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

Constraint Satisfaction Problems (CSP)

Coloração de mapas: 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



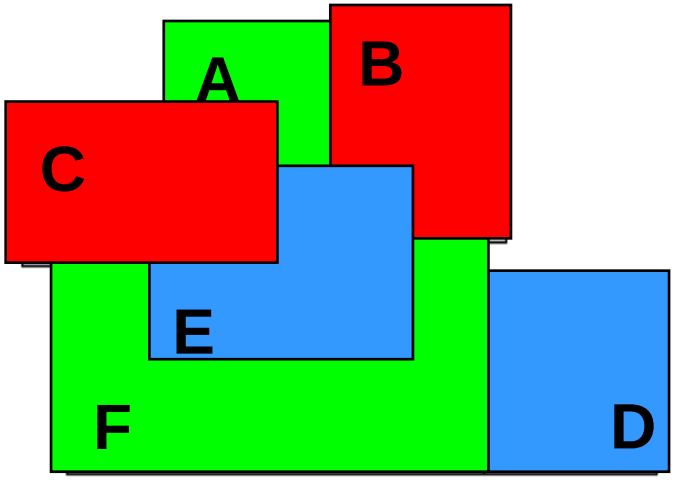
Candidatas[1]: E, F, ...resto
E = green
Candidatas: F, ...resto
F = red
Candidatos: A, B, C, D
A= red
Candidatos: B, C, D
B= blue
Candidatos: C, D
C= blue
D = green

SEM BACKTRACK!!

[1] em ordem de prioridade

Constraint Satisfaction Problems (CSP)

Coloração de mapas: variável mais restringida (variável que pode assumir menos valores)



Candidatas: todas
A = green
Candidatas: B, C, E, ...
B = red
Candidatos: E, F, ...
E=blue
Candidatos: C, F, D
C=red
Candidatos: F, D
F=green
D = blue ou red

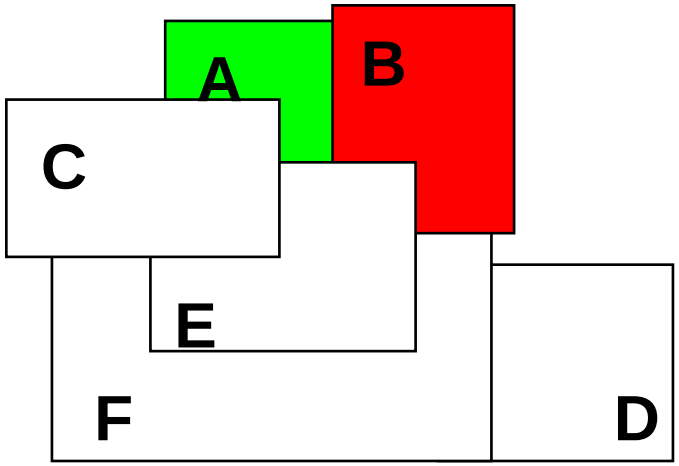
SEM BACKTRACK!!

Constraint Satisfaction Problems (CSP)

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

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

Começando com
A = green
B = red
C=??? red é melhor do que **blue**



CSP iterativo

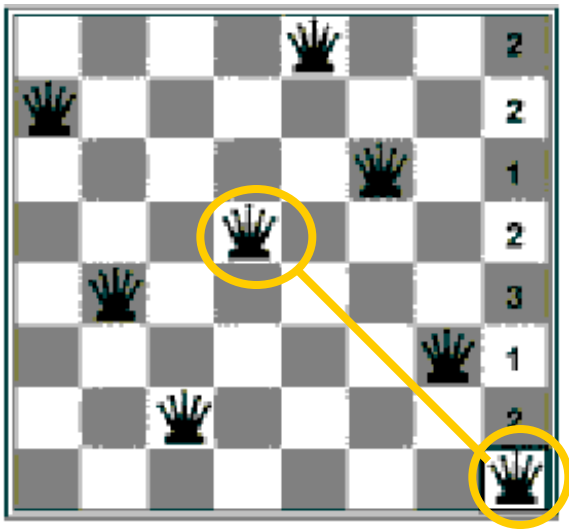
- **CSP pode ser resolvido iterativamente**
  1) instancia aleatoriamente todas variáveis;
  2) aplica operadores para trocar os valores e então diminuir número de restrições não satisfeitas (min-conflicts).

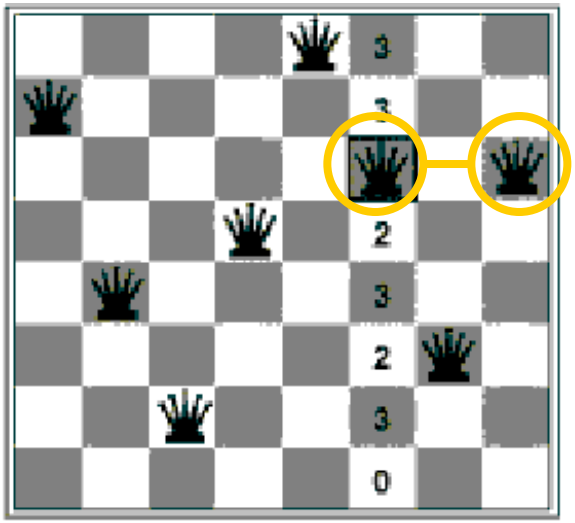
- **Heurística de reparos**
  - repara inconsistências

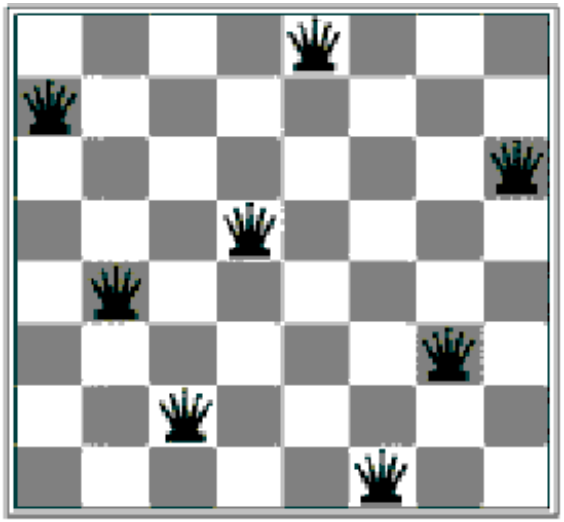
- **Min-conflict resolve 8 rainhas em menos de 50 passos!!!**

Número de ataques

- **Exemplo:**







Constraint Satisfaction Problems (CSP)

- **Grande importância prática, sobretudo em tarefas de**
  - Criação, projeto (*design*)
  - Agendamento (*scheduling*)
  - Em que várias soluções existem e é mais fácil dizer o que não se quer...