Introdução aos Agentes Inteligentes

Aula: Busca com Satisfação de Restrições
Constraint Satisfaction Problems (CSP)

Sergio Queiroz, baseado em aula de Flávia Barros

Roteiro

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

Problemas de Busca Até Então...

- Nos algoritmos de busca, cada estado é atômico, ou indivisível
  - Uma caixa preta, sem estrutura interna
- Muitas vezes é possível resolver problemas de forma mais eficiente através de representações fatoradas para os estados
  - Um conjunto de variáveis, cada qual com um valor
  - O problema estará resolvido quando cada variável tiver um valor que satisfaça todas as restrições sobre a variável.

Constraint Satisfaction Problems

- 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 consistem em:
  - um conjunto de variáveis, $\{X_1, ..., X_n\}$;
  - um conjunto de domínios, $\{D_1, ..., D_n\}$, um para cada variável.
    - Cada domínio D_i consiste em um conjunto de valores possíveis, $\{v_1, ..., v_k\}$ para a variável X_i.
  - um conjunto de restrições que especificam propriedades da solução
    - valores que essas variáveis *podem* assumir

CSP: Formulação

- Estados: definidos pelos valores possíveis das variáveis
- Estado inicial: nenhuma variável instanciada ainda
- Operadores: atribuem valores às variáveis
- 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

CSP: características das restrições

- O conjunto de valores que a variável pode assumir é chamado de 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) - ex. 8-rainhas
  - n-árias - ex. palavras cruzadas
  - 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)

Restrições: formulação

- Cada restrição C_i consiste em um par
  - <*escopo*, *rel*>
    - *escopo* é uma tupla de variáveis que participam da restrição
    - *rel* é uma relação que define os valores que essas variáveis podem assumir
      - Uma relação pode ser representada como uma lista explícita de todas as tuplas de valores que satisfazem a restrição (representação em extensão) ou uma relação abstrata (propriedade) que possui duas operações: testar se uma tupla é um membro da relação e enumerar os membros da relação (representação em intenção).

Exemplo 1

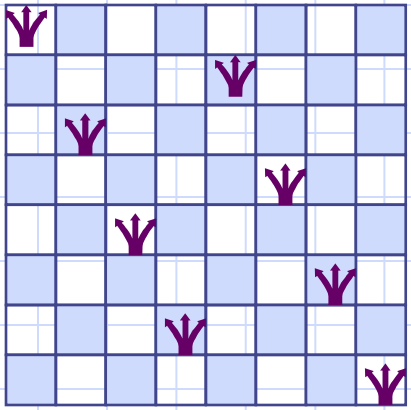
- Variáveis: X_1 e X_2.
- Domínios: $D_1 = D_2 = \{A, B\}$
- Restrições:
  - As duas variáveis devem ter valores diferentes
    - escopo: (X_1, X_2)
    - relação:
      - em extensão: $[(A, B), (B, A)]$
        - O que define a restrição $<(X_1, X_2), [(A, B), (B, A)]>$
      - em intensão: $X_1 <> X_2$.
        - O que caracteriza a restrição $<(X_1, X_2), X_1 <> X_2>$

Solução de um CSP

- Definições
  - Uma atribuição de valores a variáveis é:
    - Consistente ou legal: não viola quaisquer restrições
    - Completa: atribui valores para todas as variáveis do problema
    - Parcial: atribui valores para apenas algumas das variáveis.
- Solução para um CSP
  - Uma atribuição consistente e completa.

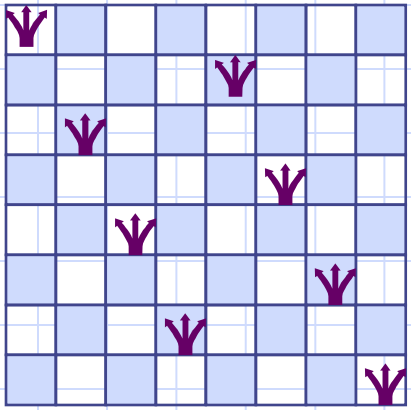
Exemplo Informal

- Jogo das 8-rainhas
  - variáveis: localização das rainhas
  - domínios das variáveis: posições do tabuleiro
  - restrições binárias: duas rainhas não podem estar na mesma coluna, linha ou diagonal
  - solução: valores para os quais a restrição é satisfeita



Exemplo Informal

- Jogo das 8-rainhas
  - variáveis: localização das rainhas
  - domínios das variáveis: posições do tabuleiro
  - restrições binárias: duas rainhas não podem estar na mesma coluna, linha ou diagonal
  - solução: valores para os quais a restrição é satisfeita



Exercício: formalize este exemplo

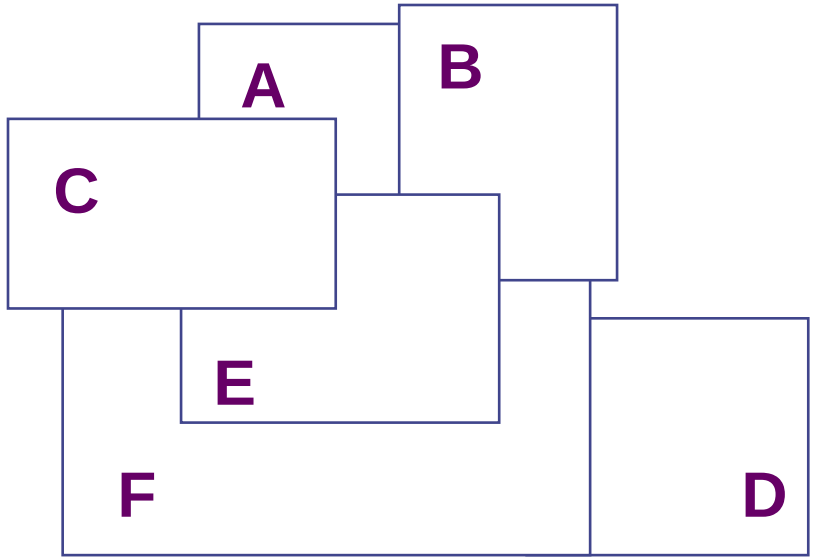
Busca cega para CSP

- Funcionamento
  - estado inicial: variáveis sem atribuição
  - aplica operador: instanciar uma variável
  - teste de parada: todas variáveis instanciadas sem violações
- Análise
  - pode ser implementada com 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

Exemplo: coloração de mapas

Simulação passo a passo...
A= green
B = green
C= green
D=green
E=green
F=green (falha c/ C, E, D)
F=red
E (falha c/ C,A,B)
E=red (falha c/ F)
E=blue
C (falha c/ A)
...
Muito dispendioso

variáveis: A,B,C,D,E,F
domínio: {green,red,blue}
restrições*: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F



* escrevemos A ≠ B como um atalho para <(A, B), A ≠ B>

Backtracking na Busca Cega

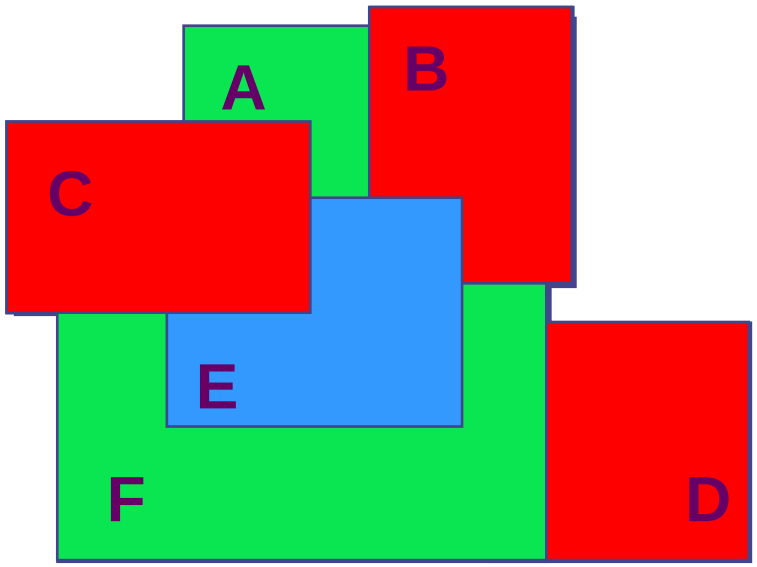
- Problema da busca em profundidade
  - perda de tempo, pois continua mesmo que uma restrição já tenha sido violada
    - não se pode mais redimir o erro
- Solução: Backtracking
  - depois de realizar uma atribuição, verifica se restrições não são violadas
  - caso haja violação ⇒ backtrack

Exemplo: coloração de mapas

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

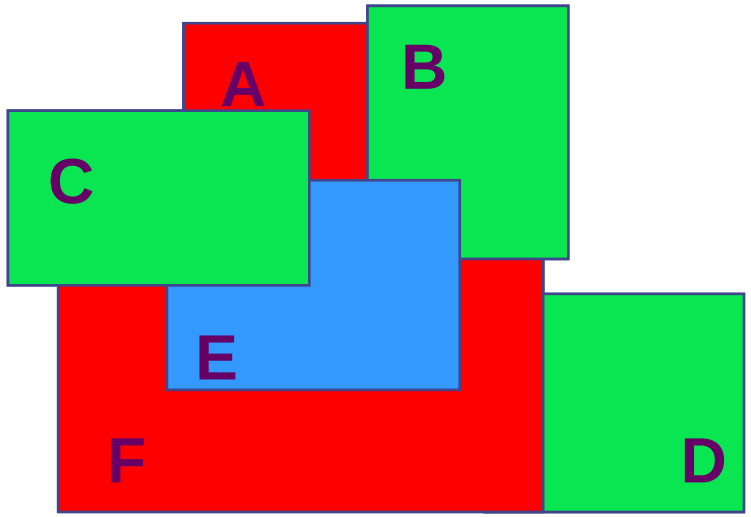
variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={green,red,blue}
restrições: $A \neq B$; $A \neq C$; $A \neq E$; $B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$; $D \neq F$; $E \neq F$



Exemplo: coloração de mapas

Mas poderia ser mais complicado começando por red...
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={green,red,blue}
restrições: $A \neq B$; $A \neq C$; $A \neq E$; $B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$; $D \neq F$; $E \neq F$



Backtracking não basta...

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

Busca Cega - Refinamentos

- Verificação prévia (*forward checking*)
  - idéia: olhar para frente para detectar situações insolúveis
    - ex. no restaurante self-service ou no bar...
- 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, *backtrack* imediatamente
- É bem mais eficiente!

Propagação de Restrições

- *Forward checking* é um caso particular de verificação de arco-consistência
  - um estado é arco-consistente se o valor de cada variável é consistente com as restrições sobre esta variável
  - arco-consistência é obtida por sucessivas eliminações de valores inconsistentes
- Propagação de restrições (*constraint propagation*)
  - uma conseqüência da verificação de arco-consistência
  - quando um valor é eliminado, outros podem se tornar inconsistentes e terem que ser eliminados também
  - é como uma onda que se propaga: as escolhas ficam cada vez mais restritas

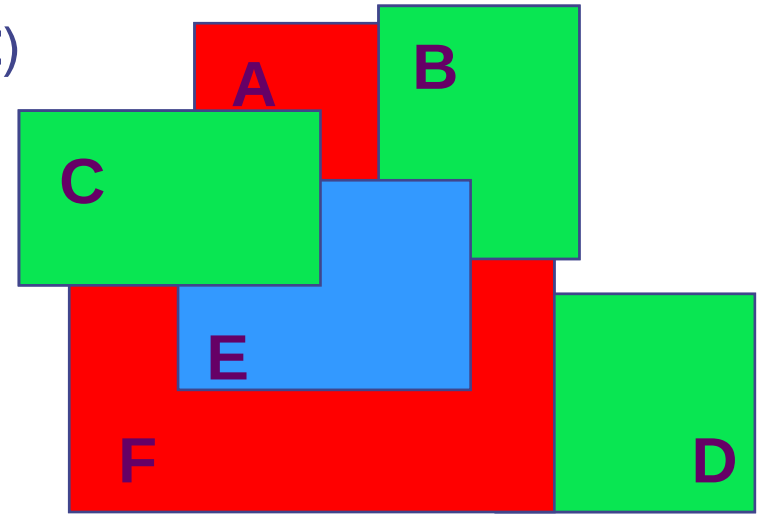
Propagação de restrições
Exemplo: coloração de mapas

variáveis: A,B,C,D,E,F
domínios ={red,green,blue}

Passo a passo...

A=red
=> B, C, E ={green,blue} (restrições c/ A)
=> D, F ={red,green,blue}
B=green
=> E = {blue}, F = {red, blue} (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!!
D=green, E=blue, F=red



Heurísticas para CSP

- Tentam 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

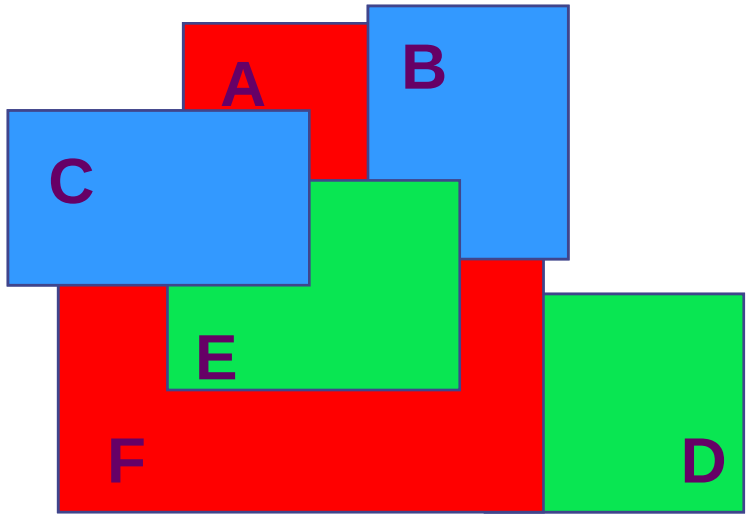
Variável mais restritiva

(variável envolvida no maior número de restrições)

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



[1] *em ordem de prioridade*

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

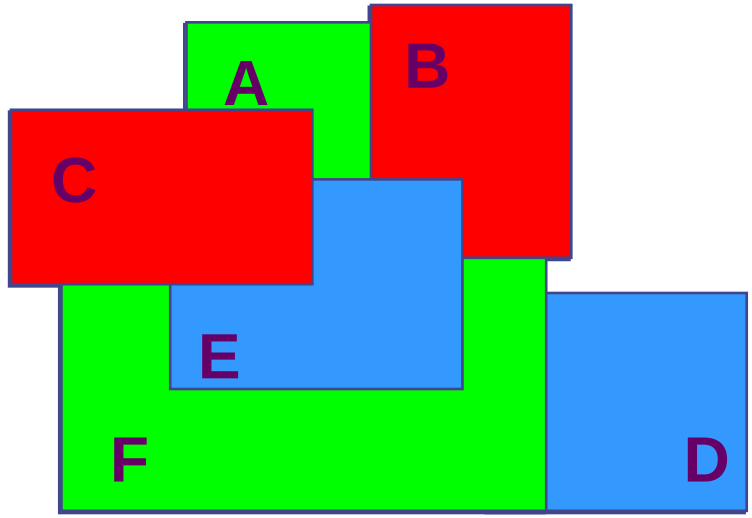
Candidatos: F, D

F=green

D = blue ou red

SEM BACKTRACK!!

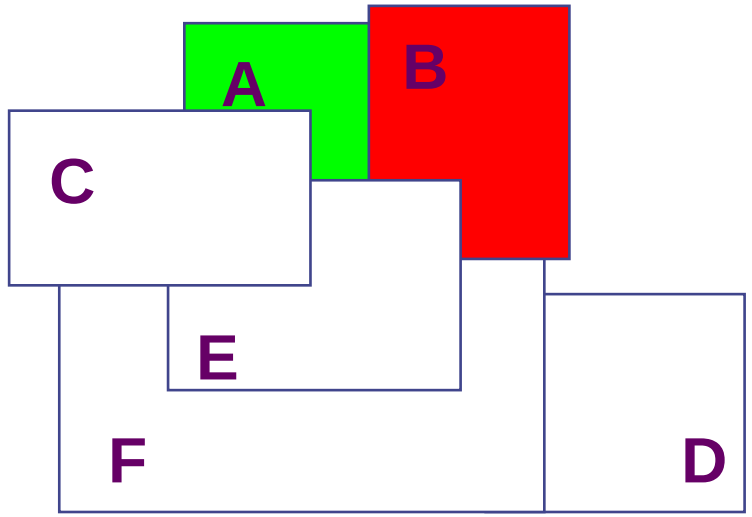
variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={green,red,blue}
restrições: $A \neq B$; $A \neq C$; $A \neq E$; $B \neq E$; $B \neq F$; $C \neq E$; $C \neq F$; $D \neq F$; $E \neq F$



Valor menos restritivo
(valor que deixa mais liberdade)

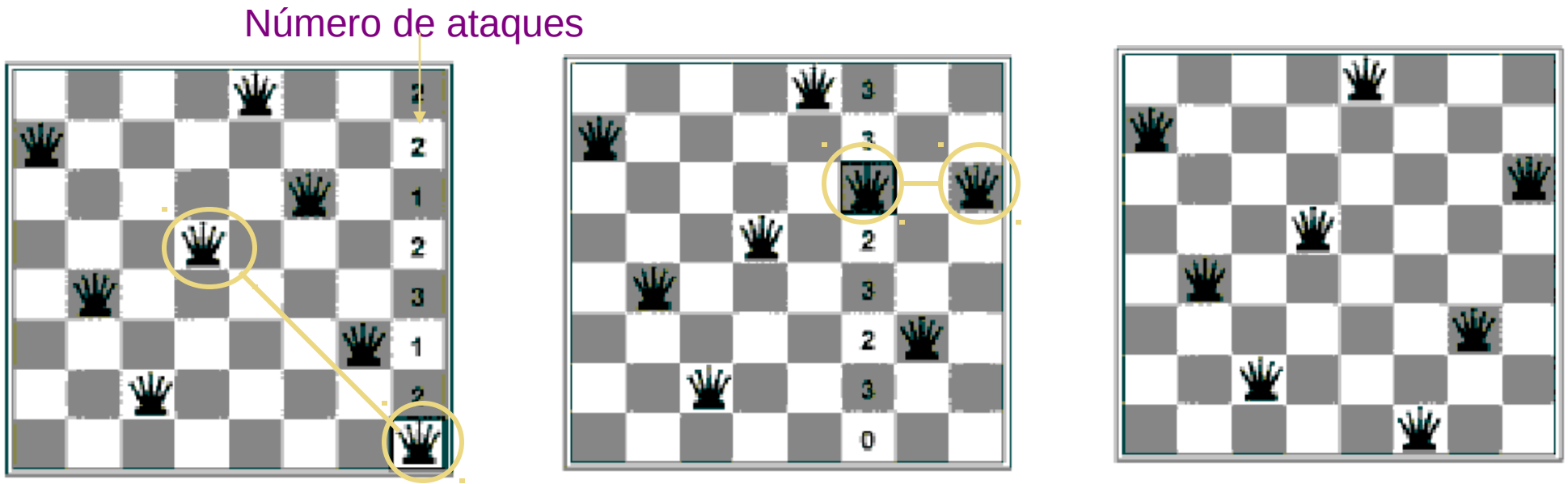
variáveis: A,B,C,D,E,F
domínio: Da=Db...=Df={green,red,blue}
restrições: A ≠ B; A ≠ C; A ≠ E; B ≠ E; B ≠ F; C ≠ E; C ≠ F; D ≠ F; E ≠ F

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!!!



CSP

- Grande importância prática, sobretudo em tarefas de
  - criação (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

A seguir...

- Até a próxima aula...
  - Lançamento da Lista de exercícios
    - Valendo 1,0 ponto da nota da 2ª. unidade
    - Cobrindo assuntos da prova
- Na próxima aula
  - Início do assunto a ser usado no projeto
    - Sistemas baseados em conhecimento