Inteligência Artificial

CTC15 Aula 3b

Sumário

- ♢ Exemplos de PSR
- ♢ Busca genérica aplicada à PSRs
- ♢ *Backtracking*
- ♢ Verificação *forward*
- ♢ Heurísticas para PSRs

Problemas de satisfação de restrições (PSRs)

Problema padrão:

Um estado é um "caixa preta"—qualquer estrutura de dados
que suporte testes de objetivo, avaliação, sucessor

PSR:

estado é definido por variáveis V_i assumindo *valores* do domínio D_i

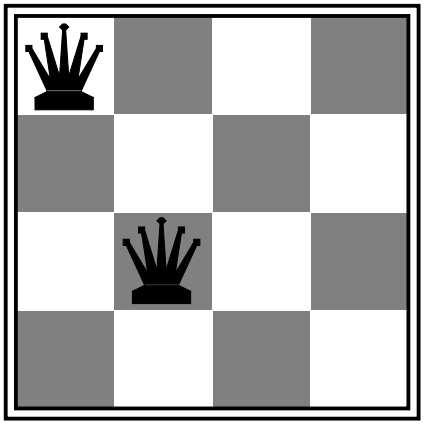
o teste de objetivo é um conjunto de restrições especificando
combinações permissíveis de valores para subconjuntos de variáveis

Exemplo simples de uma *linguagem de representação formal*

Permite o uso de algoritmos de *propósito geral* mais poderosos
que algoritmos de busca usuais

Exemplo: 4-Rainhas como um PSR

Assuma uma rainha em cada coluna. Em qual fila entra cada uma?



Variáveis Q_1, Q_2, Q_3, Q_4

Domínios $D_i = \{1, 2, 3, 4\}$

Restrições

$Q_i \neq Q_j$ (ñ pode estar na mesma coluna)

$|Q_i - Q_j| \neq |i - j|$ (ou na mesma diagonal)

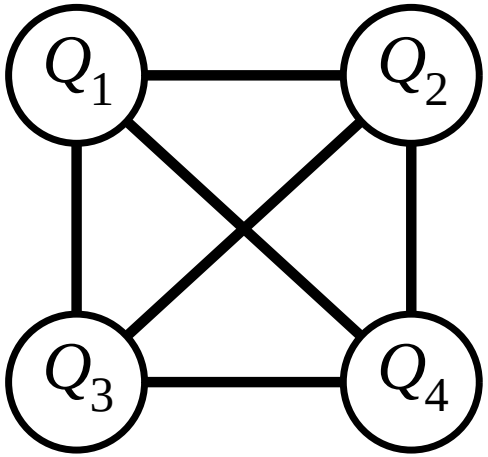
Traduza cada restrição como um conjunto de valores permissíveis para suas variáveis

Por exemplo, valores para (Q_1, Q_2) são $(1,3)$ $(1,4)$ $(2,4)$ $(3,1)$ $(4,1)$ $(4,2)$

Grafo de restrições

PSR Binário: cada restrição relaciona duas variáveis no máximo

Grafo de restrições: nós são variáveis, arcos mostram as restrições



Exemplo: Criptoaritmética

Variáveis

$D\ E\ M\ N\ O\ R\ S\ Y$

Domínios

$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Restrições

$M \neq 0$, $S \neq 0$ (restrições *unárias*)

$Y = D + E$ ou $Y = D + E - 10$, etc.

$D \neq E$, $D \neq M$, $D \neq N$, etc.

$$\begin{array}{cccccc} & \mathsf{S} & \mathsf{E} & \mathsf{N} & \mathsf{D} \\ + & \mathsf{M} & \mathsf{O} & \mathsf{R} & \mathsf{E} \\ \hline \mathsf{M} & \mathsf{O} & \mathsf{N} & \mathsf{E} & \mathsf{Y} \end{array}$$

Exemplo: Coloramento de mapa

Colorir um mapa de forma que países adjacentes não tenham a mesma cor

Variaveis

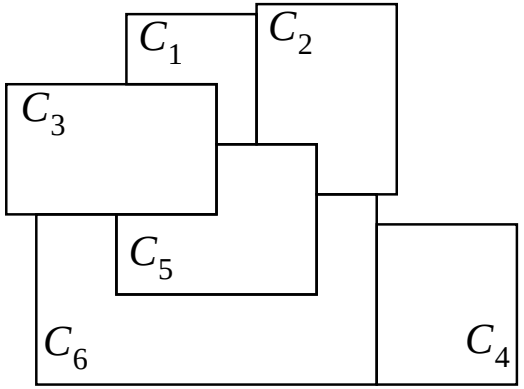
Países C_i

Domínios

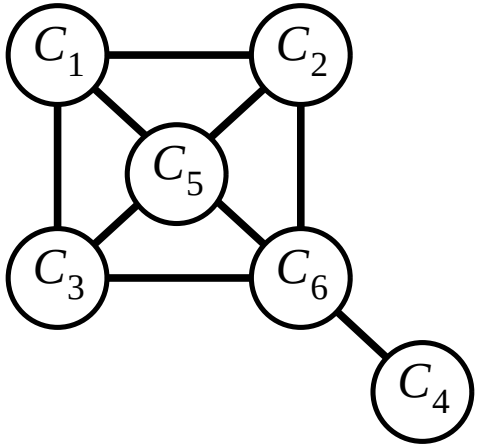
$\{Vermelho, Azul, Verde\}$

Restrições

$C_1 \neq C_2$, $C_1 \neq C_5$, etc.



Grafo de restrições:



PSRs no mundo real

Problemas de indicação
    por exemplo, quem ensina que curso

Problemas de organização
    por exemplo, qual curso é oferecido quando e onde?

Configuração de *hardware*

Planilhas

Problemas de fluxo de transporte

Planificação em fábricas

Observe que muitos problemas reais envolvem variáveis reais

Aplicando busca genérica

Comecemos com um método ingênuo, depois aperfeiçoamos

Estados são definidos pelos valores nomeados até o momento

Estado inicial: nenhuma variável nomeada

Operadores: indique um valor a uma variável não nomeada

Teste de objetivo: todas as variáveis nomeadas, nenhuma restrição violada

Note que isto ocorre para todos os PSRs!

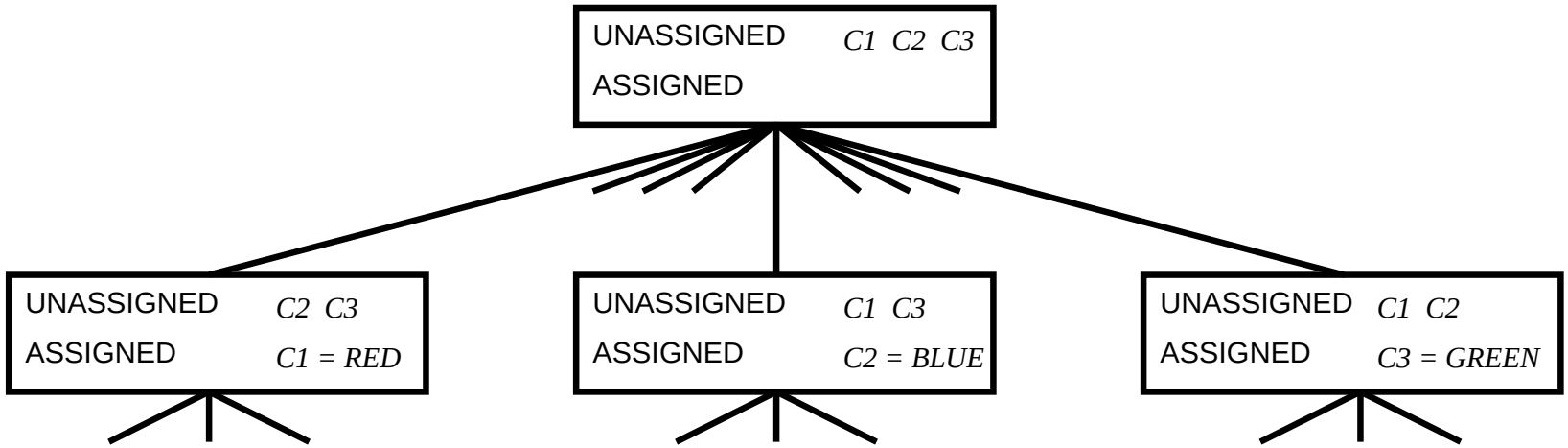
Implementação

O estado em um PSR mantém rastro de que variáveis tiveram valores nomeados até o momento
Cada variável tem um domínio e um valor atual

datatype CSP-STATE
  components: UNASSIGNED, a list of variables not yet assigned
    ASSIGNED, a list of variables that have values
datatype CSP-VAR
  components: NAME, for i/o purposes
    DOMAIN, a list of possible values
    VALUE, current value (if any)

As restrições podem ser representadas
  explicitamente como jogos de valores permissíveis, ou
  implicitamente por uma função que testa a satisfação da restrição

Busca genérica aplic. ao coloramento de mapa



Complexidade da aproximação ingênua

Profundidade max. do espaço m??

Profundidade do espaço de soluções d??

Algoritmo de busca a utilizar??

Fator de ramificação b??

Complexidade da aproximação ingênua

Profundidade max. do espaço m?? n (número de variáveis)

Profundidade do espaço de soluções d?? n (todas as vars. nomeadas)

Algoritmo de busca a utilizar?? *depth-first*

Fator de ramificação b?? $\sum_i |D_i|$ (no topo da árvore)

Isto pode ser melhorado dramaticamente observando-se o seguinte:

1) a ordem das nomeações é irrelevante, portanto muitos caminhos são equivalentes
2) as nomeações adicionadas não podem corrigir uma restrição violada

Backtracking

Uso busca *depth-first*, mas

1) fixo a ordem de nomeações, $\Rightarrow b = |D_i|$
   (pode ser feito na função SUCESSORS)
2) verifico violações de restrições

A verificação de violação de restrições pode ser implementada de duas maneiras:

1) modifico SUCESSORS para só nomear valores
   permitidos, dados os valores já nomeados ou
2) verifico se restrições são satisfeitas antes de expandir um estado

Backtracking é o algoritmo desinformado básico para PSRs

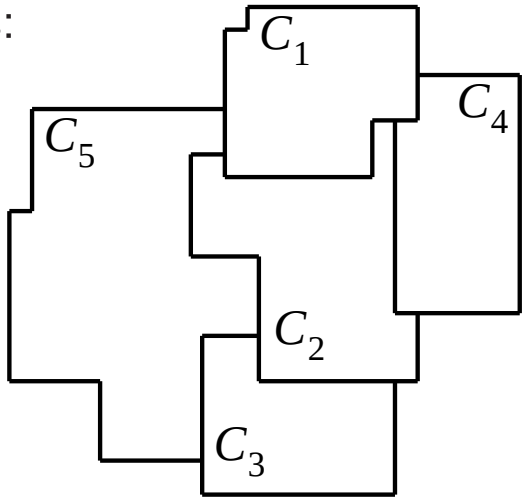
Possa resolver o n-rainhas para $n \approx 15$

Verificação forward

Idéia: Mantenha rastro dos valores legais restantes para variáveis não nomeadas
Terminar a busca quando qualquer variável não tiver nenhum valor legal

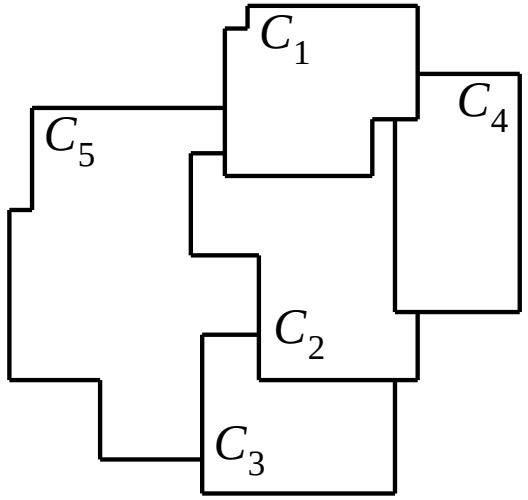
Exemplo simplificado de coloramento de mapas:

	VERMELHO	AZUL	VERDE
C_1			
C_2			
C_3			
C_4			
C_5			

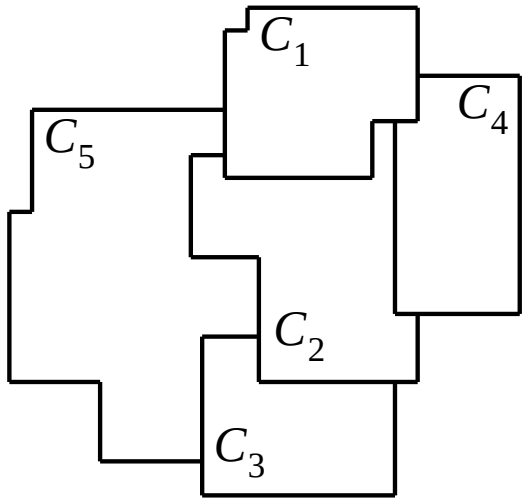


Pode resolver o n-rainhas até $n \approx 30$

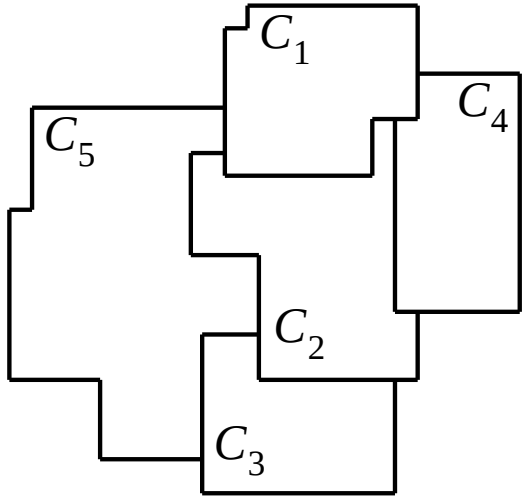
	VERMELHO	AZUL	VERDE
C_1	$\surd$		
C_2	$\times$		
C_3			
C_4	$\times$		
C_5	$\times$		



	VERMELHO	AZUL	VERDE
C_1	$\surd$		
C_2	$\times$	$\surd$	
C_3		$\times$	
C_4	$\times$	$\times$	
C_5	$\times$	$\times$	



	VERMELHO	AZUL	VERDE
C_1	$\surd$		
C_2	$\times$	$\surd$	
C_3		$\times$	$\surd$
C_4	$\times$	$\times$	
C_5	$\times$	$\times$	$\times$



Heurísticas para PSRs

Decisões mais inteligentes para

que valor escolher para cada variável

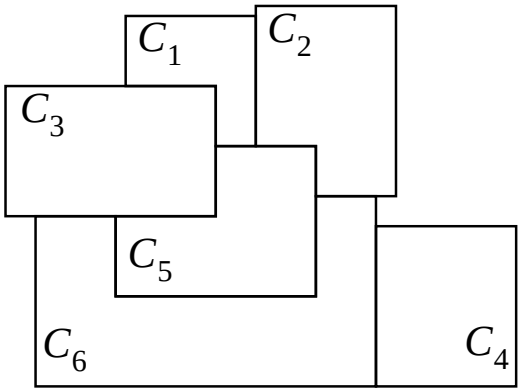
que variável nomear

dados $C_1 = Vermelho$, $C_2 = Verde$, escolha $C_3 =$??

.

dados $C_1 = Vermelho$, $C_2 = Verde$, o que fazer??

.



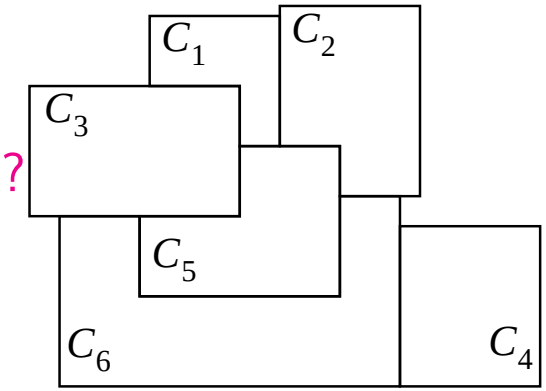
Pode resolver o n-rainhas para $n \approx 1000$

Heurísticas para PSRs

Decisões mais inteligentes para

que valor escolher para cada variável

que variável nomear



Dados $C_1 = Vermelho$, $C_2 = Verde$, escolha $C_3 =$??

$C_3 = Verde$: *valor menos restritivo*

Dados $C_1 = Vermelho$, $C_2 Verde$, que fazer??

C_5: *variável mais restritiva*

Pode resolver o n-rainhas para $n \approx 1000$

Algoritmos iterativos para PSRs

Hill-climbing e *simulated annealing* tipicamente operam com estados "completos" i.e., todas as variáveis nomeadas

Para aplicar a PSRs:

- permita estados com restrições não satisfeitas
- operadores *renomeiam* valores para variáveis

Seleção de variáveis: aleatoriamente selecione qualquer variável conflitante

Heurística de mínimos conflitos:

- escolha valor que viola menos restrições
- i.e., *hillclimb* com $h(n) =$ número total de restrições violadas

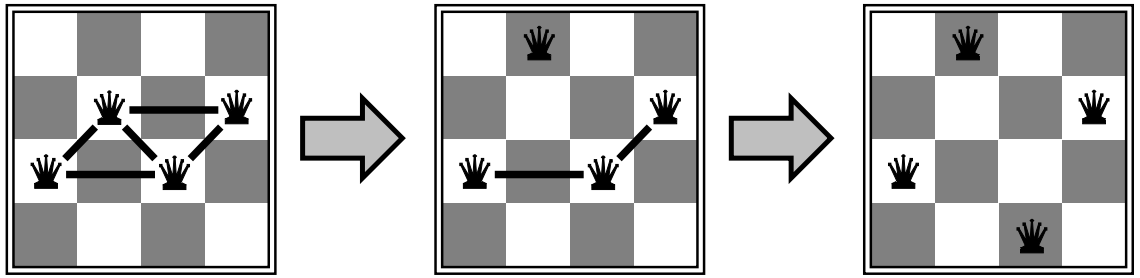
Exemplo: 4-Rainhas

States: 4 rainhas em 4 colunas ($4^4 = 256$ estados)

Operadores: mover rainha em coluna

Teste de objetivo: nenhum ataque

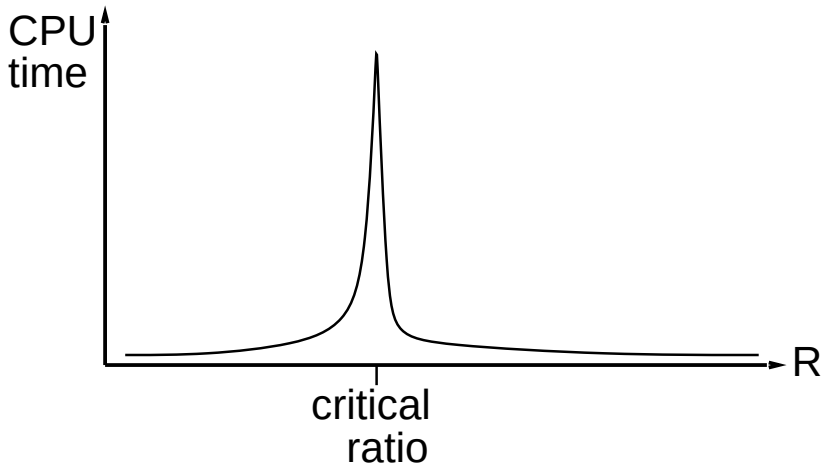
Avaliação: $h(n) =$ número de ataques



Heurística de mínimos conflitos: desempenho

Dado estado inicial aleatório, pode resolver o n-rainhas em tempo quase constante para n arbitrário, com alta probabilidade (por exemplo, $n = 10.000.000$)

O mesmo parece ser verdade para qualquer PSR aleatoriamente gerado exceto em uma faixa estreita da relação $R = \frac{\text{número de restrições}}{\text{número de variáveis}}$



Resumo

- PSRs constituem um tipo especial de problema: estados definidos por valores de um conjunto fixo de variáveis teste de objetivo definidos por *restricc oes* nos valores das variáveis
- Backtracking = busca em profundidade com:
  1. ordem fixa de variáveis
  2. só sucessores legais
- Verificação forward previne nomeações que levem a fracasso posterior
- Ordenamento de variáveis e heurística de seleção de valores ajudam significativamente
- Conflitos mínimos iterativo é normalmente efetivo na prática