Problemas e Estratégias

March 31, 2016

Exemplos de Aplicações

- Jogo dos oito :-)
- Mundo dos blocos (ex: torre de Hanoi)
- Problema das n-rainhas
- Criptoaritmética
- Missionários e Canibais
- Resta-um
- e muitos outros...

n-Rainhas

- Problema: posicionar n rainhas em um tabuleiro $n \times n$ de forma que nenhuma das rainhas se ataque nas diagonais, linhas ou colunas.
- 2 formas de se resolver o problema: incremental e completo
- Possíveis estados (a) e jogadas (b):
  1.
     - a) qq arranjo de 0 a 8 rainhas no tabuleiro
     - b) adicionar 1 rainha a qq posição do tabuleiro
  2.
     - a) arranjos de 0 a 8 rainhas com nenhuma em posição de ataque
     - b) colocar 1 rainha na próxima coluna vazia mais à esquerda de forma que não seja atacada por outra rainha
  3.
     - a) arranjos de 8 rainhas, 1 em cada coluna
     - b) mover qq rainha atacada para outra posição na mesma coluna

Criptoaritmética

```
                    SOLUCAO
  FORTY             29786      com: F = 2
+  TEN                850           O = 9
+  TEN                850           R = 7 etc
--------            ------
  SIXTY             31486
```

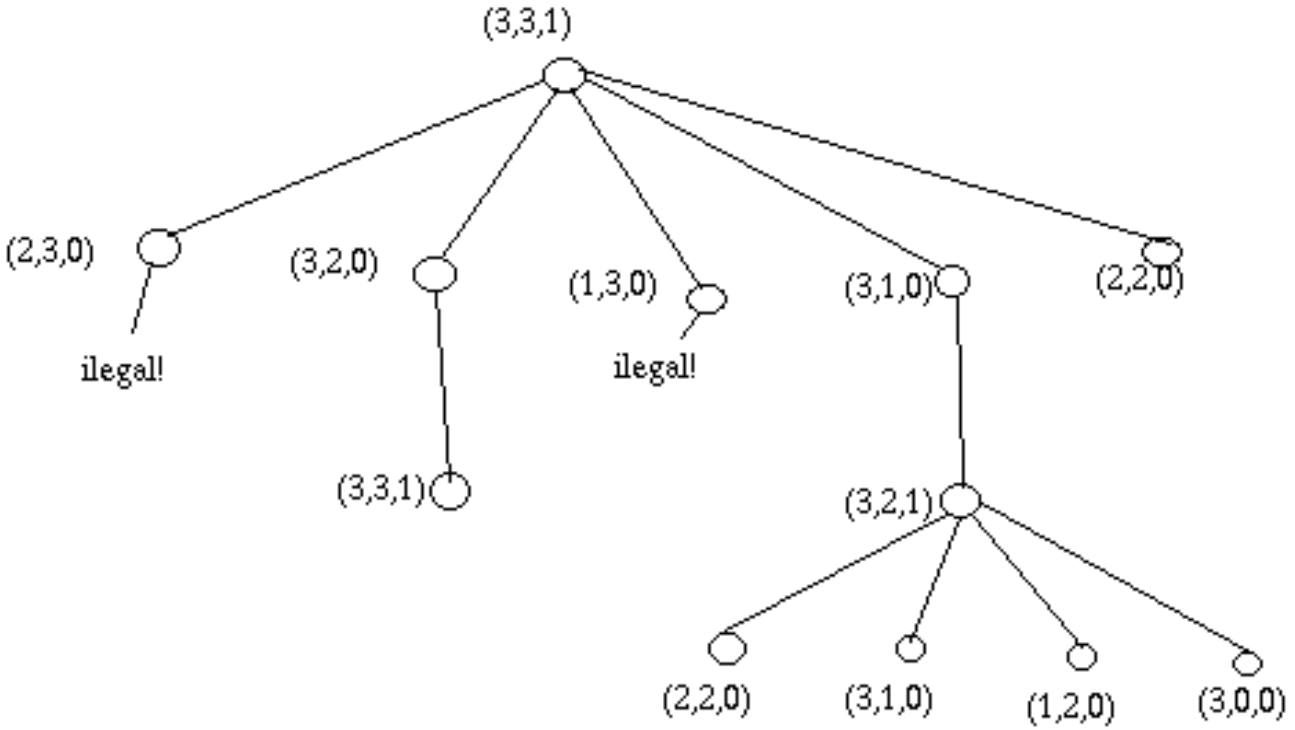
Criptoaritmética

- estados: conjunto de letras que são substituídas por dígitos
- operadores: substituir todas as ocorrências de uma letra por um dígito que ainda não tenha aparecido
- possíveis regras de seleção dos dígitos: ordem alfabética para evitar repetições, seleção mais restrita respeitando as operações aritméticas
- estado final: jogo contém somente dígitos e representa uma soma correta
- custo da solução: zero

Missionários e Canibais

- 3 missionários e 3 canibais estão na margem de um rio com um bote onde cabem, no máximo, 2 pessoas.
- Problema: encontrar uma forma de passar todos para a outra margem sem nunca deixar um grupo de missionários em uma margem com um número maior de canibais.
- estados: qq seqüência ordenada de 3 números representando o no. de missionários, canibais e botes na margem do rio.
- estado inicial: (3,3,1)
- estado final: (0,0,0)
- custo: número de travessias no rio.
- regras: tirar 1 missionário, tirar 1 canibal, tirar 2 canibais etc...

Missionários e Canibais



Sistemas de Produção

- Elementos principais de um sistema de produção em IA:
  - bancos de dados global
  - regras de produção ou restrições
  - sistema de controle

Outros Algoritmos

- Satisfação de Restrições
  - tipo especial de problema que satisfaz propriedades estruturais além dos requisitos básicos para problemas gerais de busca
  - estados: conjunto de variáveis
  - domínio: conjunto possível de valores que uma variável pode assumir (discreto/contínuo, finito/infinito)
  - estado inicial: todas as variáveis com valores possíveis iniciais
  - estado final: valores finais para as variáveis que respeitem as restrições do problema
  - profundidade máxima da árvore: número de variáveis
- Além disto: representação explícita para as restrições do problema

Satisfação de Restrições: Exemplo

- n-rainhas
  - variáveis: posições no tabuleiro
  - restrições: nenhuma rainha pode atacar outra na mesma linha, diagonal ou coluna
  - valores iniciais das variáveis: V_1 *in* $1..n$, V_2 *in* $1..n$ etc, com n largura do tabuleiro

Satisfação de Restrições: Possíveis algoritmos

- aloca uma nova rainha para uma nova coluna a cada nível (solução incremental)
- complexidade: $n^n = \approx \pi D_i = D_1 \times D_2 \times ... \times D_n$
- fator de ramificação: n
- fator máximo de ramificação:
  $n \times n = \sum D_i = D_1 + D_2 + ... + D_n$
  - se todas as variáveis fossem instanciadas com todos os valores possíveis no primeiro nível da árvore

Satisfação de Restrições: Possíveis algoritmos

```
n = 8
                        (0,0,0,0,0,0,0,0)

N  (1,0,0,0,0,0,0,0)  (0,1,0,0,0,0,0,0)  (0,0,1,0,0,0,0,0) ....
i  (2,0,0,0,0,0,0,0)  (0,2,0,0,0,0,0,0)  (0,0,2,0,0,0,0,0) ....
v   ..........
1  (8,0,0,0,0,0,0,0)  (0,8,0,0,0,0,0,0)  (0,0,8,0,0,0,0,0) ....

N  (1,1,0,0,0,0,0,0)  (0,1,1,0,0,0,0,0)  (1,0,1,0,0,0,0,0) ....
i  (2,2,0,0,0,0,0,0)  (0,2,2,0,0,0,0,0)  (2,0,2,0,0,0,0,0) ....
v   ..........
2  (8,8,0,0,0,0,0,0)  (0,8,8,0,0,0,0,0)  (8,0,8,0,0,0,0,0) ....
```

Satisfação de Restrições: Possíveis algoritmos

- utilizando BP a sub-árvore que contém

  `(0,0,0,0,0,0,0,0) -> (1,0,0,0,0,0,0,0) ->`

  `(1,1,0,0,0,0,0,0) ->`

  `(1,1,1,0,0,0,0,0) -> (1,1,1,1,0,0,0,0) -> ...`

- será explorada mesmo sabendo que (1,1,1,1,1,1,1,1) não é solução
- Classe de algoritmos "generate-and-test"

Satisfação de Restrições: Possíveis algoritmos

- solução: colocar o teste de restrição a cada rainha colocada no tabuleiro
- tb não é a melhor solução!
- Suponha que já conseguimos alocar 6 rainhas, mas esta alocação ataca a oitava rainha.
- BP testa todas as possibilidades de colocação da sétima rainha!
- Classe de algoritmos: "constrain-and-generate"

Satisfação de Restrições: Possíveis algoritmos

- solução: algoritmos *Forward Checking* e *Lookahead* ou simplesmente "consistência de arcos"
- Forward Checking: retira dos domínios de outras variáveis todos os valores impossíveis que violam as restrições
- Lookahead: além de executar forward checking, verifica se o domínio modificado de cada variável conflita com os outros.

Forward Checking: Exemplo

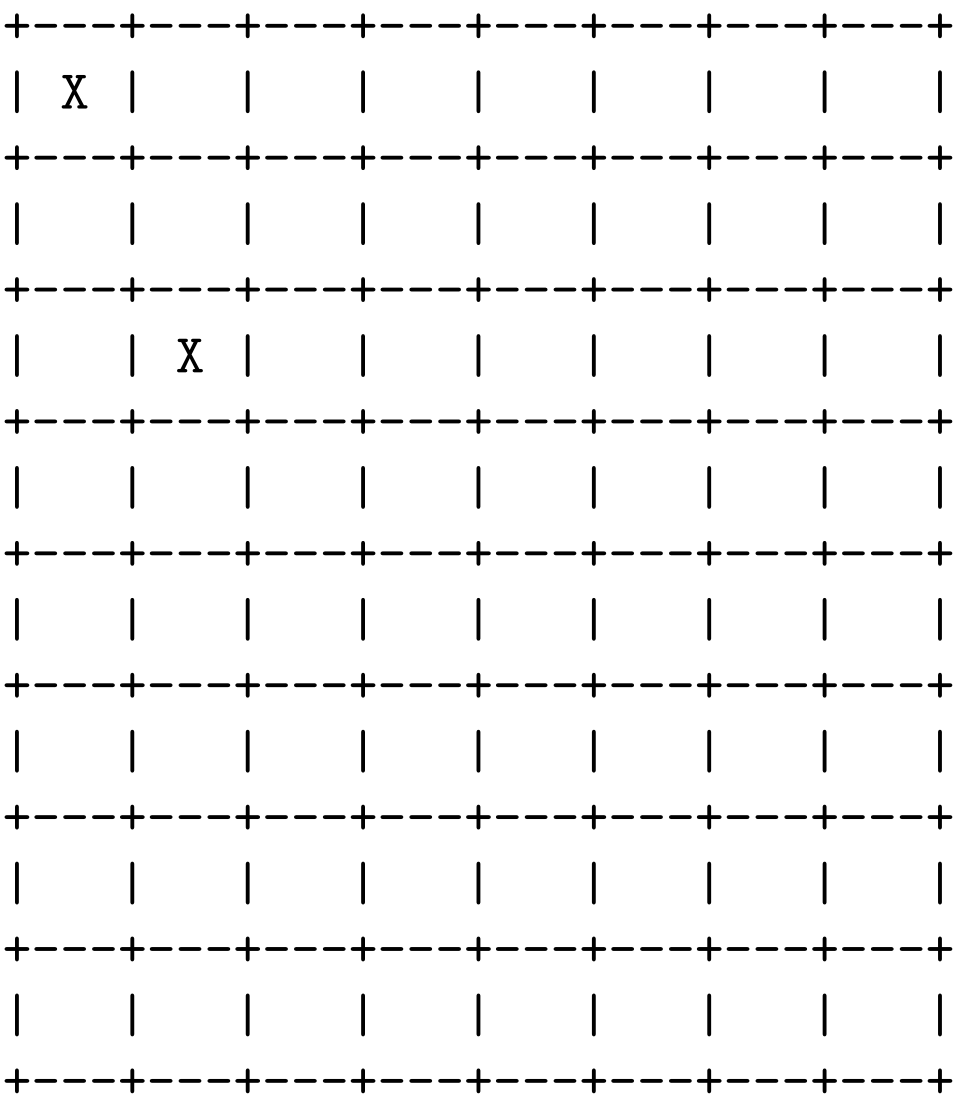
```
n = 8
(1) V1 = 1  ==> V2 = {3,4,5,6,7,8}
                V3 = {2,4,5,6,7,8}
                V4 = {2,3,5,6,7,8}
                V5 = {2,3,4,6,7,8}
                V6 = {2,3,4,5,7,8}
                V7 = {2,3,4,5,6,8}
                V8 = {2,3,4,5,6,7}

(2) V2 = 3  ==> V3 = {5,6,7,8}
                V4 = {2,6,7,8}
                V5 = {2,4,7,8}
                V6 = {2,4,5,8}
                V7 = {2,4,5,6}
                V8 = {2,4,5,6,7}
```

Forward Checking: Exemplo

n-rainhas
n=8

apos V1=1
V2=3



Forward Checking: Exemplo

- Restrições do problema:
  - $V_i = k \rightarrow V_j \neq k \quad \forall \quad j = 1, \ldots, 8; j \neq i$
  - $V_i = k_i, V_j = k_j \rightarrow \mid i - j \mid \neq \mid k_i - k_j \mid \quad \forall \quad j = 1, \ldots, 8; j \neq i$

Soluções para n-queens sem busca

(Explicit Solutions to the N-Queens Problem for all N, by Bo Bernhardsson)

- para n par e da forma $6k + 2$
  - $(j, 2j)$
  - $\frac{n}{2} + j, 2j - 1$, para $j = 1, 2, \ldots, \frac{n}{2}$
- para n par, mas não na forma 6k
  - $(j, 1 + [2(j-1) + \frac{n}{2} - 1 \mod n])$
  - $(n+1-j, n-[2(j-1)+\frac{n}{2}-1 \mod n])$, para $j = 1, 2, \ldots, \frac{n}{2}$
- para n ímpar
  - Usar qualquer um dos casos acima para n-1 e estender com uma rainha (n, n)

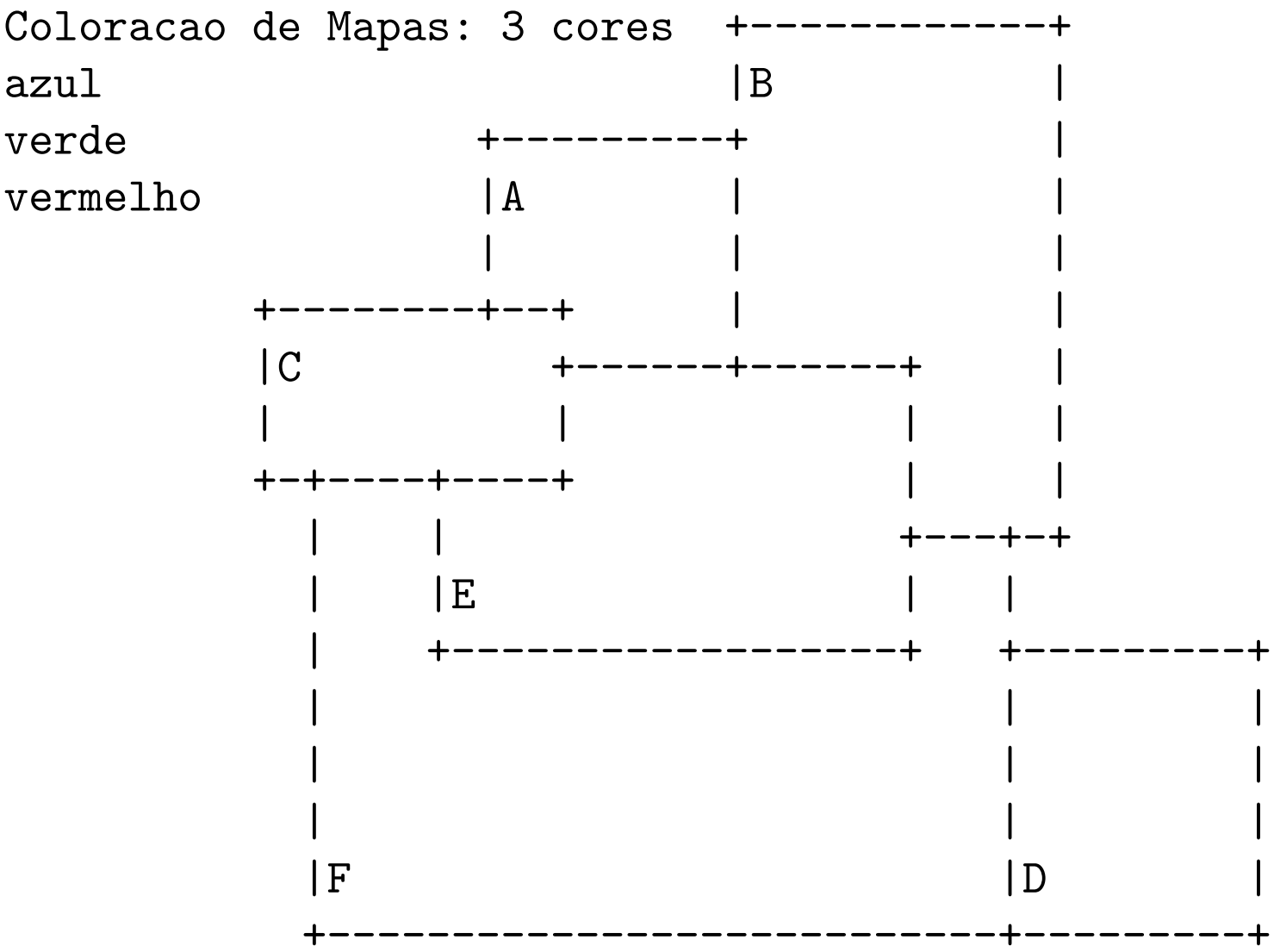
Soluções para n-queens sem busca

- Outras referências:
  - The n-Queens Problem, by Igor Rivin, Ilan Vardi, and Paul Zimmermann
  - A simplified solution of the N queens' problem, by Matthias Reichling
- Observação: encontrar **todas** as soluções para o problema das n-queens não é trivial :(

Satisfação de Restrições: Possíveis algoritmos

- para domínios infinitos: programação linear, simplex, revised simplex, convex hull, eliminação de Gauss.
- para domínios finitos: forward checking, lookahead, consistência de arcos em geral.
- Para domínios finitos, dois problemas:
  - escolha da *variável*:
    - *most-constrained*: menor domínio
    - *most constraining*: restringe ao máximo os domínios das outras variáveis
  - escolha do *valor* da variável:
    - princípio *first-fail*.
    - *least constraining*: valor que afeta menos o conjunto de valores das outras variáveis

Forward Checking: Exemplo



Satisfação de Restrições

- variável most-constrained: permite resolver n-rainhas com n igual a 100.
- forward checking puro: só consegue resolver até 30.
- valor least-constraining: permite resolver n-rainhas com n igual a 1000.

Algoritmos

- Backtracking (busca sistemática)
- Propagação de Restrições (constraint propagation: k-consistency)
- Os que usam heurísticas para ordenação de variáveis e valores de variáveis
- Backjumping e dependency-directed backtracking

Algoritmo básico

```
CSP-BACKTRACKING(PartialAssignment a)
     If a is complete then return a
     X <- select an unassigned variable
     D <- select an ordering for the domain of X
     For each value v in D do
         If v is consistent with a then
              Add (X = v) to a
              result <- CSP-BACKTRACKING(a)
              If result <> failure then return result
              Remove (X = v) from a
     Return failure
```

Chamada inicial: CSP-BACKTRACKING()
Resolve n-queens para $n \approx 25$
Combinação de constraint propagation com heurísticas para a escolha da próxima variável e escolha do próximo valor de variável consegue resolver 1000-queens.