

Universidade Federal de Ouro Preto  
BCC 325 - Inteligência Artificial  
Raciocínio com Restrições

Prof. Rodrigo Silva

## 1 Leitura

- Capítulo 4 do livro *Artificial Intelligence: Foundations of Computational Agents, 3rd Edition*, de David L. Poole e Alan K. Mackworth. Disponível em <https://artint.info/>

## 2 Questões

1. (Seção 4.1.1) O que é uma variável em um problema de satisfação de restrições (CSP)? Diferencie variáveis discretas de variáveis contínuas.
2. (Seção 4.1.1) O que é uma atribuição total de variáveis? Quantas atribuições totais existem para um CSP com  $n$  variáveis, cada uma com domínio de tamanho  $d$ ?
3. (Seção 4.1.2) Defina e exemplifique:
  - (a) Restrição unária;
  - (b) Restrição binária;
  - (c) Restrição ternária;
  - (d) Definição intencional e extensional de uma restrição.
4. (Seção 4.2) Considere o CSP com variáveis  $A, B, C \in \{1, 2, 3, 4\}$  e restrições  $A < B$  e  $B < C$ :
  - (a) Quantas atribuições totais existem?
  - (b) Quantas dessas atribuições satisfazem todas as restrições?
  - (c) Esboce a árvore de busca até o nível em que ocorre a primeira poda por violação de restrição.
5. (Seção 4.1.1) Qual é a vantagem de modelar estados por meio de variáveis em vez de enumerar todos os estados explicitamente?
6. (Seção 4.2) Explique o funcionamento do algoritmo `DFS_solver` da Figura 4.1. Por que a verificação de restrições em atribuições parciais pode reduzir significativamente o espaço de busca?
7. (Seção 4.1.3) Considere o seguinte problema de agendamento de entregas com as variáveis  $A, B, C, D, E$ , todas com domínio  $\{1, 2, 3, 4\}$  e as seguintes restrições:
$$\{(B \neq 3), (C \neq 2), (A \neq B), (B \neq C), (C < D), (A = D), (E < A), (E < B), (E < C), (E < D), (B \neq D)\}$$
  - (a) Encontre uma solução que satisfaça todas as restrições;
  - (b) Justifique quais atribuições puderam ser descartadas por poda antes de explorar totalmente a árvore de busca.
8. (Reflexiva) Descreva uma situação do mundo real (diferente dos exemplos dados no livro) que pode ser modelada como um CSP. Especifique as variáveis, seus domínios e pelo menos duas restrições.

9. (Seção 4.3) Consistência de arco e GAC:
- (a) Defina o que significa um arco  $\langle X, c \rangle$  ser *consistente de arco* (arc-consistent) em relação ao dicionário de domínios  $\text{dom}$ .
  - (b) O que significa uma rede de restrições ser *arc-consistent*?
  - (c) Descreva os três possíveis estados em que a rede pode se encontrar após a execução do algoritmo **GAC** (Figura 4.4): comente o que cada caso implica sobre a existência e unicidade de soluções.
10. (Seção 4.3, Exemplo 4.14/4.18) Reconsidere o CSP da Questão 4, com variáveis  $A, B, C \in \{1, 2, 3, 4\}$  e restrições  $A < B$  e  $B < C$ .
- (a) Desenhe a rede de restrições (constraint network) correspondente, indicando nós de variáveis, nós de restrições e arcos  $\langle X, c \rangle$ .
  - (b) Considere o conjunto inicial  $to\_do$  contendo todos os arcos da rede. Aplique manualmente o algoritmo **GAC** (Figura 4.4), apresentando uma tabela com:
    - o arco selecionado em cada passo;
    - se houve ou não redução de domínio;
    - quais novos arcos foram adicionados a  $to\_do$ .
  - (c) Indique os domínios finais de  $A, B$  e  $C$  após o término do algoritmo. Compare com os domínios iniciais e discuta em que sentido o problema foi “simplificado” para a busca em profundidade.
11. (Seção 4.3, Exemplo 4.20) Considere o CSP com variáveis  $A, B$  e  $C$ , cada uma com domínio  $\{1, 2, 3, 4\}$ , e restrições  $A = B$ ,  $B = C$  e  $A \neq C$ .
- (a) Mostre que essa rede é arc-consistent, isto é, que nenhum domínio pode ser reduzido usando apenas uma restrição por vez.
  - (b) Mostre que, apesar disso, o CSP não possui solução (não há atribuição que satisfaça todas as restrições simultaneamente).
  - (c) Explique por que esse exemplo mostra que a consistência de arco, por si só, não é suficiente para decidir a satisfatibilidade de um CSP em geral. Comente brevemente como técnicas de consistência de ordem superior (por exemplo, path consistency) podem ajudar nesse tipo de situação.

### 3 Problema Prático

#### 1. Backtracking para Sudoku e análise experimental

Considere o problema de resolver jogos de Sudoku  $9 \times 9$  usando busca com backtracking.

##### (a) Implementação do resolvedor

Implemente um algoritmo de *backtracking* para resolver instâncias de Sudoku. Sua implementação deve:

- representar o tabuleiro como uma matriz  $9 \times 9$  (ou estrutura equivalente);
- escolher, em cada passo, uma célula ainda não preenchida;
- testar valores possíveis naquela célula, verificando se a atribuição é consistente com as regras do Sudoku (linha, coluna e subgrade  $3 \times 3$ );
- retroceder (backtrack) quando não houver valor possível para a próxima célula.

(b) **Contagem de atribuições avaliadas**

Modifique seu código para contar o *número de atribuições avaliadas até que a primeira solução seja encontrada*. Para isso:

- considere como *atribuição avaliada* cada tentativa de atribuir um valor a uma célula ainda não preenchida (mesmo que essa tentativa venha a ser rejeitada logo em seguida por violar alguma restrição);
- incremente um contador toda vez que o algoritmo tentar colocar um valor em uma célula durante a busca.

Ao final da execução, o algoritmo deve imprimir (ou retornar) o número total de atribuições avaliadas para resolver aquela instância.

(c) **Seleção das instâncias**

Escolha, em fontes confiáveis na internet (por exemplo, sites de Sudoku):

- 5 instâncias classificadas como “fáceis”;
- 5 instâncias classificadas como “difíceis” (por exemplo, “hard”, “expert” ou equivalente).

Para cada instância, registre:

- o identificador ou link da fonte;
- a classificação de dificuldade fornecida pelo site.

(d) **Experimento**

Para cada uma das 10 instâncias (5 fáceis e 5 difíceis):

- execute o seu algoritmo de backtracking;
- registre o número de atribuições avaliadas até a primeira solução ser encontrada.

Organize os resultados em uma tabela, separando claramente os grupos de Sudokus fáceis e difíceis.

(e) **Análise estatística**

- Calcule a **média** e o **desvio padrão** do número de atribuições avaliadas para o conjunto de problemas fáceis.
- Calcule a **média** e o **desvio padrão** do número de atribuições avaliadas para o conjunto de problemas difíceis.

Apresente os resultados (por exemplo, em uma pequena tabela) e escreva um parágrafo discutindo:

- se, no seu experimento, os Sudokus difíceis exigiram mais atribuições avaliadas que os fáceis;
- possíveis razões para diferenças observadas (por exemplo, estrutura dos puzzles, ordem de escolha de variáveis, heurísticas usadas ou não, etc.).