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.

3 Problemas Práticos

1. Implemente o algoritmo A* para o problema do labirinto.
2. Considere o seguinte problema de alocação de salas:

   Uma universidade precisa alocar um conjunto de disciplinas em salas e horários disponíveis, respeitando restrições como:

   - Uma sala só pode ter uma aula por horário;
   - Um professor só pode dar uma aula por vez;
   - A capacidade da sala deve ser suficiente para o número de alunos da disciplina;
   - Algumas disciplinas exigem salas específicas (por exemplo, laboratórios).

   (a) Modele este problema como um problema de satisfação de restrições (CSP), identificando:
   - As variáveis do problema;
   - Os domínios de cada variável;
   - As restrições envolvidas.

   (b) Escreva um código baseado em busca com backtracking que resolva esse problema, seguindo a lógica dos exemplos de Sudoku ou N-Rainhas vistos em sala. Considere o exemplo abaixo como caso de teste.

   Dados de Entrada:

```
aulas = ['Aula1', 'Aula2', 'Aula3']

dominios = {
    'Aula1': [('Sala1', '08h'), ('Sala2', '08h'), ('Sala1', '10h')],
    'Aula2': [('Sala1', '08h'), ('Sala2', '10h')],
    'Aula3': [('Sala2', '08h'), ('Sala2', '10h'), ('Sala1', '14h')]
}

constraints = {
    'Aula1': {'professor': 'ProfA', 'alunos': 30, 'sala_requerida': None},
    'Aula2': {'professor': 'ProfB', 'alunos': 20, 'sala_requerida': 'Sala2'},
    'Aula3': {'professor': 'ProfA', 'alunos': 35, 'sala_requerida': None}
}

salas = {
    'Sala1': {'capacidade': 40},
    'Sala2': {'capacidade': 25}
}
```

   Saída:

```
[
    {'aula': 'Aula1', 'sala': 'Sala1', 'horario': '08h', 'professor': 'ProfA'},
    {'aula': 'Aula2', 'sala': 'Sala2', 'horario': '10h', 'professor': 'ProfB'},
    {'aula': 'Aula3', 'sala': 'Sala1', 'horario': '14h', 'professor': 'ProfA'}
]
```