Roteiro de Aula – Problemas de Satisfação de Restrições (CSPs)

Disciplina: Inteligência Artificial

1 Definição

Um **Problema de Satisfação de Restrições** (*Constraint Satisfaction Problem* – CSP) é definido por:

- Um conjunto de **variáveis** $X = \{X_1, X_2, \ldots, X_n\}$;
- Um **domínio** de valores possíveis para cada variável D_i;
- Um conjunto de **restrições** $C = \{C_1, C_2, \ldots, C_m\}$, que especificam combinações permitidas de valores.

O objetivo é encontrar uma atribuição de valores a todas as variáveis que **satisfaça todas as restrições**. Formalmente:

$$A : X_i \to D_i \quad \text{tal que todas as } C_j \text{ sejam verdadeiras.}$$

2 Exemplos de Problemas Clássicos

Coloração de Mapas: variáveis = regiões; domínio = {vermelho, verde, azul}; restrição: regiões adjacentes não podem ter a mesma cor.

N-Rainhas: variáveis = colunas; domínio = linhas; restrição: nenhuma rainha pode atacar outra.

Sudoku: variáveis = células da grade; domínio = {1,...,9}; restrição: sem repetição em linhas, colunas e blocos.

3 Exemplos de Problemas Práticos

- **Agendamento de tarefas (Scheduling):** alocar disciplinas, professores e salas sem sobreposições.
- **Planejamento de rotas (Roteirização):** planejar entregas respeitando janelas de tempo e capacidade.
- **Design de circuitos / layout industrial:** posicionar componentes evitando sobreposição.

4 Algoritmo *Generate and Test*

Ideia: gerar todas as combinações possíveis e testar quais satisfazem as restrições.

Etapas

1. Gerar uma atribuição completa (ou parcial) de valores;
2. Testar se todas as restrições são satisfeitas;
3. Se sim, retornar a solução; caso contrário, tentar outra combinação.

Pseudocódigo

```
for cada combinacao poss vel de valores:
    if satisfaz_todas_as_restricoes():
        return solucao
return "sem solucao"
```

Limitação: ineficiente — cresce exponencialmente com o número de variáveis e domínios.

5 Backtracking

A técnica de **backtracking** é um aperfeiçoamento do *generate and test* que evita explorar soluções sabidamente inválidas.

Ideia

Construir a solução incrementalmente, voltando atrás (*backtrack*) quando uma restrição é violada.

Etapas

1. Atribuir valor à primeira variável;
2. Testar restrições parciais;
3. Se houver violação, voltar à variável anterior e tentar outro valor;
4. Repetir até que todas as variáveis estejam atribuídas ou as possibilidades se esgotem.

Pseudocódigo

```
def Backtrack(atribuicao):
    if todas_variaveis_atribuidas(atribuicao):
        return atribuicao
    X = escolher_variavel_nao_atribuida()
    for v in dominio(X):
        if consistente(X, v, atribuicao):
            atribuicao[X] = v
            resultado = Backtrack(atribuicao)
            if resultado != "falha":
                return resultado
            del atribuicao[X]
    return "falha"
```

Vantagens

- Reduz consideravelmente o espaço de busca;
- Permite heurísticas como *Forward Checking* e *Arc Consistency*.