Mauricio Ferreira de Araujo - 222007021
Disciplina: Inteligência Artificial

UnB

Portfólio 2 - Projeto utilizando CSPs

1 Introdução

Este documento tem como foco apresentar o problema a ser resolvido, a implementação e os resultados de um projeto utilizando CSPs, desenvolvido na linguagem Python, para a disciplina de Inteligência Artificial.

Foram utilizados a IDE pycharm para implementar o algoritmo de backtraking.

2 Problema

Problemas de Satisfação de Restrições (CSPs) são problemas onde você precisa encontrar uma solução que obedeça a um conjunto estrito de regras atribuidas como restrições.

O algoritmo de backtracking se torna ideal para qualquer problema de busca combinatória, onde o número total de possibilidades é astronômico, tornando a força bruta (testar todas as combinações) impraticável, assim como o ambiente escolhido que é o jogo de sudoku.

3 Projeto utilizando CSPs desenvolvido: backtracking

O algoritmo de Backtracking é um método clássico de Inteligência Artificial a melhor forma de entendê-lo é imaginando que você está resolvendo um labirinto muito complexo você chega a uma bifurcação (no nosso caso uma celula do sudoku vazia) e precisa escolher um caminho (tentar um número dentro do dominio do sudoku) você segue esse caminho e continua. Se em algum momento você bater em um beco sem saída (o número quebrou uma regra do Sudoku ou levou a uma situação futura impossível), você deve retroceder fazendo o "backtrack"exatamente até a bifurcação anterior, lá você marca aquele caminho como errado apaga o 1 voltando a célula para 0 e tenta o próximo caminho disponível (tenta o número "2"). Esse processo se repete até que você encontre a saída (o tabuleiro fechado com o sudoku) ou tenha testado todos os caminhos e possa concluir que não há saída.

3.1 Exemplo de uso

O backtracking foi implementado de forma recursiva onde a função principal resolverSudoku chama a si mesma ela funciona como o explorador, primeiro ela chama encontrarVazio para achar a próxima bifurcação (célula com 0) se encontrarVazio não achar nada (retorna None), significa que o labirinto foi resolvido o tabuleiro está completo e a função retorna True (Sucesso).

Se uma célula vazia é encontrada, o algoritmo entra em um loop para testar todos os caminhos (os números de 1 a 9). Para cada número, ele usa a função juiz com nome de

ehSeguro para verificar se aquele caminho é válido (se o número não viola as restrições de linha, coluna ou bloco), se não for seguro, o loop simplesmente tenta o próximo número.

Se o número for seguro, o algoritmo faz uma aposta, ele coloca o número na célula e em seguida, chama resolverSudoku novamente dentro da propria função para resolver o resto do tabuleiro a partir desse novo estado. Se essa chamada recursiva eventualmente retornar True, significa que a aposta foi boa e a saída foi encontrada, então o True é propagado de volta.

Porém, se a chamada recursiva retornar False, significa que a aposta levou a um beco sem saída em algum ponto futuro, é aqui que o backtrack acontece o algoritmo desfaz a aposta, limpando a célula de volta para 0, e o loop continua, tentando o próximo número, se o loop testar todos os números de 1 a 9 e nenhum deles levar a uma solução, a função retorna False, sinalizando para a chamada anterior que ela fez uma escolha errada, forçando-a a retroceder também.

```
--- Tabuleiro Original (9x9) ---
+-------+-------+-------+
| 5 3 . | . 7 . | . . . |
| 6 . . | 1 9 5 | . . . |
| . 9 8 | . . . | . 6 . |
|-------+-------+-------|
| 8 . . | . 6 . | . . 3 |
| 4 . . | 8 . 3 | . . 1 |
| 7 . . | . 2 . | . . 6 |
|-------+-------+-------|
| . 6 . | . . . | 2 8 . |
| . . . | 4 1 9 | . . 5 |
| . . . | . 8 . | . 7 9 |
+-------+-------+-------+

...Resolvendo...

--- Solução Encontrada (em 0.0405 segundos) ---
+-------+-------+-------+
| 5 3 4 | 6 7 8 | 9 1 2 |
| 6 7 2 | 1 9 5 | 3 4 8 |
| 1 9 8 | 3 4 2 | 5 6 7 |
|-------+-------+-------|
| 8 5 9 | 7 6 1 | 4 2 3 |
| 4 2 6 | 8 5 3 | 7 9 1 |
| 7 1 3 | 9 2 4 | 8 5 6 |
|-------+-------+-------|
| 9 6 1 | 5 3 7 | 2 8 4 |
| 2 8 7 | 4 1 9 | 6 3 5 |
| 3 4 5 | 2 8 6 | 1 7 9 |
+-------+-------+-------+
```

Figura 1: Execução do algoritmo backtracking