Universidade Do Vale do Rio Dos Sinos - UNISINOS

Programa de pós-graduação em computação aplicada

Discentes: Gustavo Zanatta Bruno & Guilherme Falcão Silva Campos

Docente: Patrícia A. Jaques Maillard

Introdução

Esse trabalho tem como objetivo desenvolver uma abordagem prática com a implementação dos algoritmos de busca cega e busca heurística para resolução do problema combinatório do Sudoku. Para implementação dele optou-se pela utilização da linguagem python devido a afinidade dos membros do grupo por ela.

O trabalho consiste em modelar este problema como um problema de busca e implementar um programa que encontre uma solução para a grade 9x9 dada. Foram utilizados dois algoritmos, primeiramente a busca em Largura e depois uma implementação baseada no Busca em Largura com uma heurística baseada no a\*

O programa terá como entrada um arquivo texto que conterá uma grade 9x9 com a descrição do jogo. Exemplo do conteúdo do arquivo de entrada:

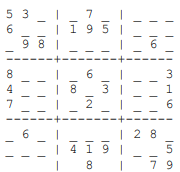


Figura 1 - Entrada proposta no enunciado do trabalho.

Desenvolvimento

A implementação do primeiro algoritmo que usa a implementação de busca em largura foi baseado no algoritmo presente em [Cormen et al. 2009]. A busca em largura a partir da raíz, no caso este se refere ao estado inicial que é o quadro com o problema inicial do Sodoku, a partir dessa raiz é construída a arvore: cada nó expandido é acrescentado à árvore e se refere a um novo estado criado a pós a expansão; cada estado é uma tentativa de adicionar um valor no quadro. Em uma fila são colocados os nós que já foram visitados, mas têm vizinhos ainda não numerados. O processo iterativo basicamente consiste no seguinte:

enquanto a fila não estiver vazia

retire um nó N da fila

para cada vizinho V de N

se V não foi um estado validado

se V é um estado com um resultado válido

retorna V

senão ponha V na fila

A implementação da heurística basicamente adiciona uma prioridade na fila na hora da adição da tentativa de preenchimento, ao invés de ir preenchendo as casas vazias em sequência, são preenchidas primeiramente a casa que têm mais valores preenchidos em suas linhas, colunas e quadrantes. Consequentemente iniciando o preenchimento a partir da casas com mais valores preenchidos em seus respectivos quadrantes, linhas, colunas a chances de erros de preenchimentos são menores.

Conclusão

Para o problema do sudoku abaixo as abordagens tiveram o seguinte desempenho:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Busca em profundidade** | **Backtracking** | **Backtracking MRV** |
| **Passos** | 20724 | 4208 | 51 |
| **Tempo (s)** | 18.29 | 0.19 | 0.07 |
| **Complexidade** | *O(n³)* | O(n²) | O(n) |

Como pode ser observado no quadro acima a implementação com o algoritmo de busca em profundidade que faz uma abordagem baseado na metodologia da busca cega obteve um tempo de resolução em torno de 20 segundos e executou 20.724 passos. A implementação que utiliza o algoritmo backtracking simples demorou 0.15 segundos e 40208 passos e por fim, a implementação que utiliza o backtracking com a heurística do menor valor restante obteve 0.07 segundos e 51 passos.

Para o problema do sudoku proposto solução que implementa o backtracking com a heurística do menor valor restante é solução é ótima, pois na primeira executou somente um passo para cada nodo vazio.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Louis, S. (2013). Artificial Intelligence. https://www.cse.unr.edu/~sushil/class/ai/notes/wk10.pdf, [accessed on Jul 14].

Mahdavipanah, H. (2017). Solve Sudoku with Python using the CSF approach. https://github.com/mahdavipanah/SudokuPyCSF, [accessed on Jul 14].