Universidade Do Vale do Rio Dos Sinos - UNISINOS

Programa de pós-graduação em computação aplicada

Discentes: Gustavo Zanatta Bruno & Guilherme Falcão Silva Campos

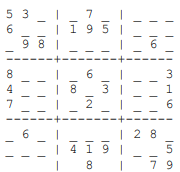
Docente: Patrícia A. Jaques Maillard

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 1 algoritmo de busca cega e 1 algoritmo de busca heurística para resolver o problema (total=2 algoritmos).

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:



Desenvolvimento

Para desenvolvimento do trabalho foi proposto a utilização de 3 algoritmos, o primeiro é o da busca em profundidade, o segundo é a implementação convencional do *backtracking* e por fim foi implementado novamente o *backtracking,* mas com a heurística do menor valor restante.

A implementação do primeiro algoritmo que usa a implementação de busca em profundidade é a seguinte [Cormen et al. 2009; Zietz 2019]:

* Ao sair do nós raiz avança um passo na busca;
* Se o nó for preenchido, avança novamente passo;
* Se o nó estiver vazio tenta adicionar um número válido entre 0 e 9;
* Se nenhum nó for válido retorne ao posicionamento do nó anterior, desfazendo a ação anterior;
* Se na ação anterior foi inserido um número menor que 9, incremente esse número;
* Se na ação foi inserido o número 9 indica que o nó anterior também era inválido, insira 0 no nó anterior e retorne novamente;

A implementação do segundo algoritmo que usa a implementação simples do backtracking é a seguinte [Cormen et al. 2009; Mahdavipanah 2017]:

* Cria uma função recursiva que utiliza uma matriz de sudoku.
* Verifique se há um local não atribuído. Se presente, atribua um número de 1 a 9, verifique se atribuir o número ao índice atual torna a grade válida ou não; se válida, chame recursivamente a função para todos os casos seguros de 0 a 9. Se alguma chamada recursiva retornar verdadeira, finalize o loop e retorne verdadeiro. Se nenhuma chamada recursiva retornar verdadeiro, retorne falso, neste caso o sudoku não tem solução;
* Se não houver um local não atribuído, retorne verdadeiro.

A implementação do terceiro algoritmo que usa uma implementação do backtracking com a heurística do menor valor restante é a seguinte [Cormen et al. 2009; Louis 2013; Mahdavipanah 2017]:

* É a mesma abordagem do algoritmo anterior com um incremento;
* Ao invés de atribuir valores simplesmente incrementando o valor inserido nó anterior é verificado previamente se a variável a ser inserida se ela não está presente no domínio de verificação (linha, coluna, bloco 3x3) ao qual o nó a ser preenchido pertence;

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].

Zietz, M. (2019). Brute-force (depth-first search) Sudoku solver in Python. https://github.com/zietzm/dfs-sudoku-solver, [accessed on Jul 14].