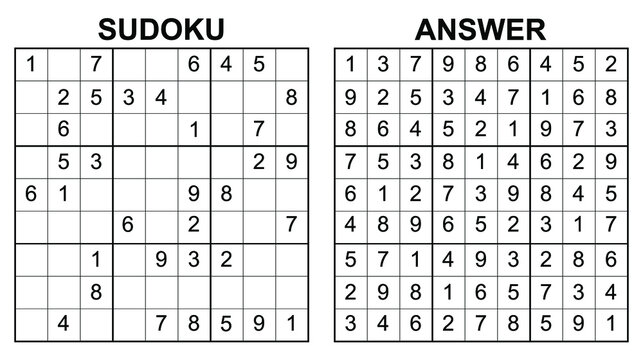
Sudoku Solver

Martin Rode, Marvin Üürike

Introduction

Sudoku is a popular number puzzle game that originated in Japan. The objective of the game is to fill a 9x9 grid with digits from 1 to 9 in such a way that each column, each row, and each of the nine 3x3 subgrids (also known as "boxes") contains all the digits from 1 to 9 without repetition. The puzzle starts with some cells pre-filled with numbers, and the player must fill in the remaining cells following these rules.

In this project, we aim to develop an AI Sudoku solver using the Depth-First Search (DFS) algorithm combined with various logical constraints. The goal is to enhance the solver's efficiency and accuracy in providing solutions to any valid Sudoku puzzle and to compare the effectiveness of different logical constraints in speeding up the solving process.



Strategies

Basic:

In this strategy, the solver inserts numbers from 1 to 9 into cells row by row, trying each possibility until a solution is found or backtracking occurs if a conflict is encountered. It's a straightforward approach where the solver systematically explores possible solutions without any optimization. This method is simple but may lead to inefficient exploration of the search space.

Original Legal Values:

Before solving the puzzle, the solver calculates the legal values for each cell based on the initial puzzle configuration and stores them in a cache. During the solving process, the solver only considers these pre-computed legal values for each cell. This reduces the search space compared to the basic approach but does not adapt to changes made during solving.

Updating Legal Values with Cell Prioritization:

This strategy involves dynamically updating the cache of legal values for each cell after every entry. Additionally, cells with the fewest legal values are prioritized for exploration. By continuously updating the cache of legal values and strategically prioritizing cells, this method can significantly improve solving efficiency, especially for hard sudokus.

Hidden Singles:

After inserting a value into a cell, the solver updates the cache of legal values for other cells affected by this entry. Additionally, it checks each block to see if the inserted value remains the only legal option for any cell within the block. If so, the excessive legal options for that cell are removed from the cache, accelerating subsequent iterations. Hidden singles represent efficient opportunities for progression in solving the puzzle, providing definitive solutions without the need for further exploration or backtracking. This mechanism can be thought of as a deep update of the cache, optimizing the solving process.

Priority Value Search:

This strategy prioritizes trying the legal values of a cell based on their occurrence frequency in the legal values of other cells within the same row, column, or block. Starting with the least occurring values increases the likelihood of making progress quickly. By prioritizing values with lower occurrence frequencies, the solver strategically explores fewer common possibilities first, potentially leading to faster solutions.

Results