Solving Sudoku using Hill Climbing

Arsenii Pogodin
ČVUT–FIT
pogodars@fit.cvut.cz

April 13, 2024

1 Problem Description

Given an incomplete Sudoku puzzle where some of the cells are filled with digits and others are left empty, the task is to fill in the empty cells while ensuring that the completed puzzle satisfies the Sudoku constraints.

2 Hill Climbing Algorithm

Hill climbing is a local search algorithm that iteratively improves a candidate solution by making incremental changes. In the context of solving Sudoku, the hill climbing algorithm works as follows:

1. Initialization: Start with a randomly filled Sudoku board.
2. Neighbor Generation: Generate a neighboring solution by making a small change to the current solution.
3. Evaluation: Calculate the score of the neighbor solution based on the duplicate digits in rows, columns, and subgrids.
4. Move Selection: If the neighbor solution has a lower score, move to that solution. Otherwise, continue with the current solution.
5. Termination: Repeat steps 2-4 until a solution score of 0 is found or restart the algorithm if a certain number of iterations are done.

3 Common Problems

While hill climbing is a straightforward algorithm, there are some common challenges when applying it to Sudoku:

- Local Optima: Hill climbing can get stuck in local optima where no better solution is reachable from the current state.
- Exploration vs. Exploitation: Hill climbing tends to exploit the current best solution rather than exploring other possibilities. This can lead to premature convergence to suboptimal solutions.

4 Used Algorithms

Three versions of neighbor generators have been implemented:

RandPosFill Randomly select a position from the cells that need to be filled with a number and then generate a random number to fill it.

BestPosFill Randomly select a position from the cells that need to be filled with a number. Then, fill it with a number that minimizes the number of conflicts in the current row and column.

SwapTwoRandom Randomly select two positions from the cells that need to be filled with a number and then swap them.

5 Results

The SwapTwoRandom strategy proved less effective than RandPosFill and BestPosFill. This is due to its reliance on purely random starting states, ignoring collision counts. Consequently, random cell swaps do little to solve the puzzle, failing even on easier ones with fewer than 10 missing values.s

Therefore, I conducted tests using only the RandPosFill and BestPosFill strategies on 9x9 and 16x16 Sudoku grids with different numbers of empty cells. Please refer to the table below for the summarized outcomes.

- **15/25 Missing Values**: Quite challenging, requires logical reasoning and careful deduction.
- **30/35 Missing Values**: Very challenging, needs advanced techniques like X-wing and swordfish.
- **40/45 Missing Values**: Extremely difficult, demands a deep understanding of Sudoku strategies and perseverance.

Strategy	Average Time for 9x9 Board		
	25 Empty Cells	35 Empty Cells	40 Empty Cells
RandPosFill	7.64 seconds (5 runs)	158.6 seconds (5 runs)	Did not solve
BestPosFill	0.41 seconds (5 runs)	11.5 seconds (5 runs)	340 seconds (3 runs)

Table 1: Average Time for Each Strategy on 9x9 Board

Strategy	Average Time for 16x16 Board		
	15 Empty Cells	30 Empty Cells	45 Empty Cells
RandPosFill	0.20 seconds (5 runs)	12.2 seconds (5 runs)	91 seconds (5 runs)
BestPosFill	0.08 seconds (5 runs)	0.4 seconds (5 runs)	0.98 seconds (5 runs)

Table 2: Average Time for Each Strategy on 16x16 Board