Sudoku Solver Report

Introduction

The Sudoku Solver uses a combination of the Arc Consistency Algorithm 3 (AC-3) and depth-first search (DFS) with Backtracking to solve a Sudoku puzzle. The solution also uses Minimum Remaining Value (MRV) heuristic and Forward Checking for further performance improvement [1]. As a requirement, all puzzles must be solved under 30 seconds each on the test machine to count as successful, but the expected result is an average of under a second per puzzle.

Methodology

The Sudoku Solver was developed and improved iteratively, adding new techniques until the best performance and accuracy was achieved. Initially, only depth-first search (DFS) was implemented, but it performed poorly on several puzzles defined in the *sudoku.ipynb*.

Afterward, the backtracking strategy was also included in the solution. That allows the algorithm to explore multiple options and "undo" a previous decision if it leads to an invalid state or a dead end. For further improve the code's performance, Minimum Remaining Value (MRV) heuristic and Forward Checking were also implemented. They allow the algorithm to make more informed decisions when searching for a possible solution.

The AC-3 algorithm has a worst-case time complexity of $O(ed^3)$ and space complexity of O(e), where e is the number of arcs and d is the size of the largest domain [2]. As such, this is also the overall complexity of the given Sudoku Solver.

Results

The Sudoku Solver solves all 60 puzzles defined in the *sudoku.ipynb* notebook under 30 seconds ranging from very easy to hard definitions. The tests were executed with average of 0.11s, per puzzle in a MacBook Pro, with Processor 2,3 GHz Dual-Core Intel Core i5, 8 GB 2133 MHz LPDDR3 of RAM memory, and Operational System (OS) macOS Ventura 13.0.1 (22A400). Only one puzzle, the 15th of the hard ones, takes more than 1s to be resolved. As a result, this meets the criteria initially established.

Conclusion

In conclusion, solving Sudoku puzzles through a brute-force approach is simple but not practical, especially for the hardest ones. A smarter algorithm that uses backtracking and heuristics such as MRV, forward checking, and arc consistency can solve even complex puzzles quickly. However, manually writing constraints is required. Possible improvements to the solution include supporting other formats besides 9x9 sudokus, the use of other data structures like Sparse Matrices, using advanced algorithms like Dancing Links (DLX) [3], genetic algorithms, or parallelizing techniques, as well as incorporating machine learning strategies like reinforcement learning.

References

[1] Russel, S., Norvig, P., 2003. *Artificial Intelligence: A Modern Approach*. 4th ed. pp.164-187.

[2] AC-3 algorithm [Online]. Available from: https://en.wikipedia.org/wiki/AC-3_algorithm [Accessed 10 January 2023].

[3] Stanford Lecture: Don Knuth - "Dancing Links" (2018) [Online]. Available from https://www.youtube.com/watch?v=_cR9zDlvP88 [Accessed 16 January 2023].