Sudoku Solver Report

March 28, 2025

This solver is based on **backtracking with constraint propagation**, a recognized method for solving Constraint Satisfaction Problems (CSPs) (Bitner et al., 1976; Dechter, 1992; Haralick & Elliott, 1980). Sudoku is represented as a CSP where variables represent grid cells, domains are digits 1–9, and constraints establish uniqueness in rows, columns, and 3x3 blocks.

A basic method like Depth-First Search (DFS) can be very slow because it might try up to $O(9^m)$ combinations, where m is the number of empty cells. To solve puzzles faster, optimizations were implemented. The algorithm prunes the search space early by eliminating values that violate constraints before attempting assignments. Additionally, it ensures that partial solutions remain valid at each recursive step.

Core Algorithm

Before solving begins, the algorithm checks if the initial Sudoku board is valid and consistent with the game's rules. This avoids working on puzzles that are already unsolvable. An extra step ensures that every empty cell has at least one valid value before solving begins.

The solver uses a recursive backtracking algorithm that fills empty cells by trying numbers from 1 to 9 in row-major order (left to right, top to bottom). Before placing a number, the algorithm checks that it doesn't break any Sudoku rules. If valid, it places the number and continues; if not, it backtracks.

This approach incorporates forward checking (verifying constraint satisfaction before assignment) and early pruning (stopping search paths that cannot lead to a solution), which together serve as basic constraint propagation and improve efficiency.

Optimisations

- **Forward Checking:** The algorithm verifies the validity of each number before placing it, avoiding conflicts in rows, columns, or sub-grids (Dechter, 1992).
- **Early Pruning:** If a cell has no valid values, the algorithm abandons that path immediately to avoid unnecessary recursion (Haralick & Elliott, 1980).

Reflections and Future Work

The solver performs efficiently across all difficulty levels. Very easy puzzles are consistently solved in under 0.02 seconds, while easy and medium puzzles are also handled swiftly, typically within 0.02 seconds. As expected, execution time increases for hard puzzles, with most cases solved in under 15 seconds, and some taking up to 30 seconds due to the exponential growth of the search space. All tested puzzles were solved correctly, with 15/15 correct solutions for each difficulty level. The solver also handles inconsistencies and unsolvable puzzles gracefully, returning -1 when appropriate.

In addition to the approach implemented in this solver, other possible strategies have been explored in the literature. These include variable selection heuristics such as Minimum Remaining Values (MRV), degree heuristics, and least-constraining-value (Dechter, 1992; Kumar, 1992; Russell & Norvig, 2021; Simonis, 2005). Constraint propagation methods like arc-consistency algorithms (e.g., AC-3), as well as alternative paradigms such as depth-first iterative deepening or hybrid techniques with lookup tables (Korf, 1985; Norvig, 2012), also represent interesting directions.

Overall, the solver delivers accurate results across all tested scenarios and performs efficiently even on complex instances. These outcomes highlight that a backtracking-based approach, when combined with basic constraint techniques, can serve as a highly effective and dependable solution for solving Sudoku puzzles.

References

Bitner, J. R., Ehrlich, G., & Reingold, E. M. (1976). Backtracking: Programming techniques. *Communications of the ACM*.

Dechter, R. (1992). Constraint networks. *Artificial Intelligence*, *57*(1), 29–57.

Haralick, R. M., & Elliott, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, *14*, 263–313.

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109.

Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, *13*(1), 32–44.

Norvig, P. (2012). Solving every sudoku puzzle [Accessed: 2025-03-28].

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Simonis, H. (2005). Sudoku as a constraint problem. *CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems*.