Sudoku Solver and Generator: Algorithms and Analysis

Your Name

December 24, 2024

Contents

1 Introduction

Sudoku is a popular number puzzle where the goal is to fill a 9x9 grid so that each row, column, and 3x3 subgrid contains the numbers 1 through 9 exactly once. Solving Sudoku puzzles computationally is classified as an NP-complete problem. This report explores the implementation of algorithms for solving and generating Sudoku puzzles, focusing on backtracking, forward checking, and AC-3 constraint propagation.

2 Data Structures Used

- **2D Arrays:** Used to represent the Sudoku grid.
- **Lists:** Manage arcs in the AC-3 algorithm.
- **Dictionaries:** Track variable assignments and domains for forward checking and AC-3.

3 Algorithms

3.1 Backtracking Solver

The backtracking algorithm is a recursive approach that iterates through empty cells, attempting to assign values that satisfy Sudoku constraints. If a value leads to a dead-end, the algorithm backtracks to try a different value.

3.2 Forward Checking

Forward checking enhances backtracking by maintaining a list of possible values for each cell (domains). When a value is assigned, it removes conflicting values from the domains of neighboring cells, reducing the search space.

3.3 AC-3

AC-3 (Arc Consistency Algorithm 3) enforces consistency between variables by iteratively checking and revising domains. If a domain becomes empty, the puzzle is unsolvable.

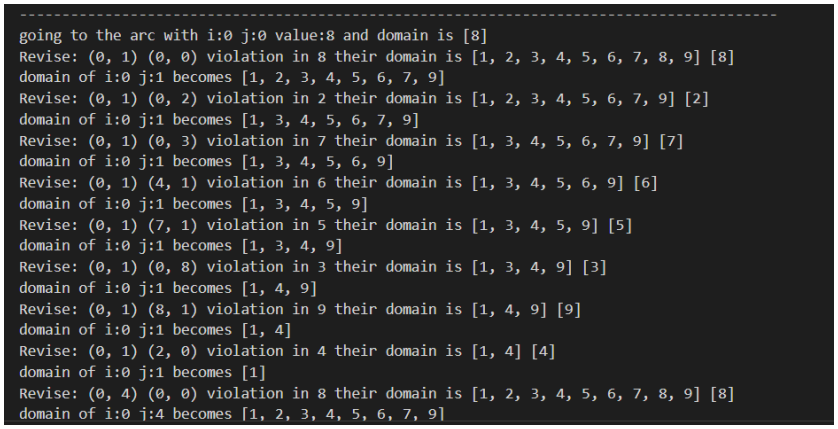

Figure 1: Arc consistency

3.4 Puzzle Generator

The generator creates a complete Sudoku board using backtracking, then removes values randomly while ensuring a unique solution.

4 Assumptions

- Input puzzles are valid 9x9 grids with zeros representing empty cells.
- At least one and unique solution exists for each puzzle.
- The generator ensures a unique solution.

5 Sample Runs

5.1 Initial and Solved Boards

5.2 Performance Metrics

Difficulty	Time (ms)	Removed Cells
Easy	20	25
Medium	27	35
Hard	30	45

Table 1: Performance Metrics for Different Difficulties

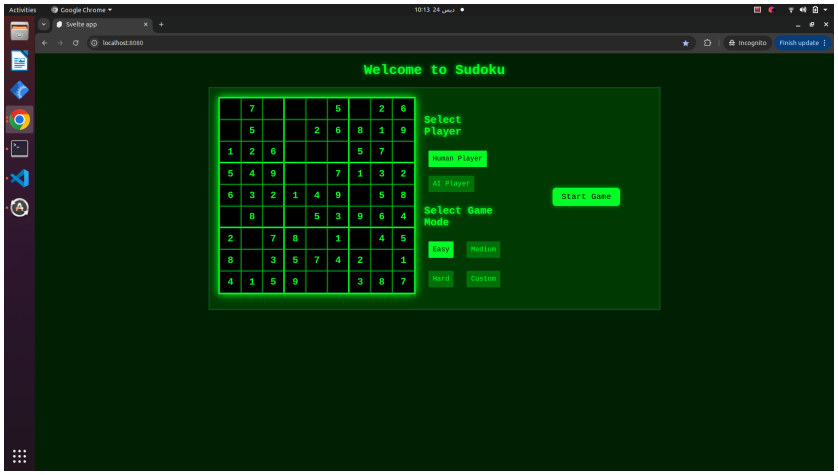

Figure 2: Initial Sudoku Board Easy

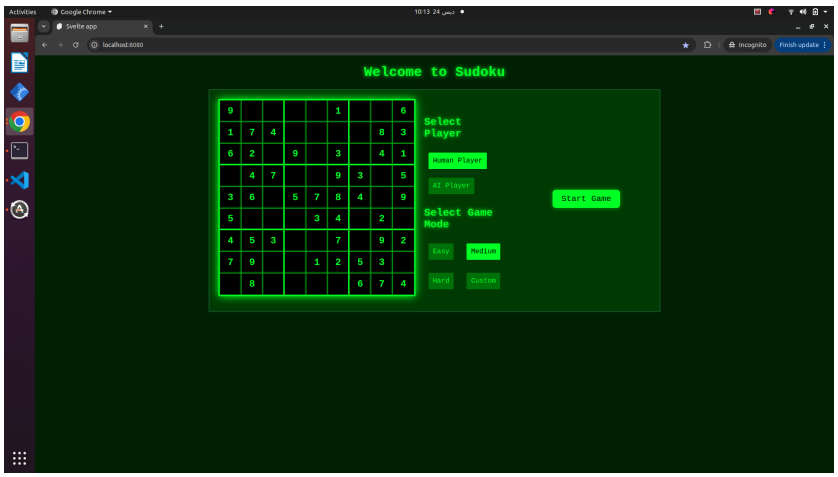

Figure 3: Initial Sudoku Board Medium

6 Comparison Between Initial Boards

Boards with fewer clues require more recursive calls and domain revisions, as they have larger initial search spaces. Hard puzzles demonstrate the effectiveness of AC-3 in reducing constraints.

7 Observations

- Forward checking significantly reduces the search space compared to plain backtracking.

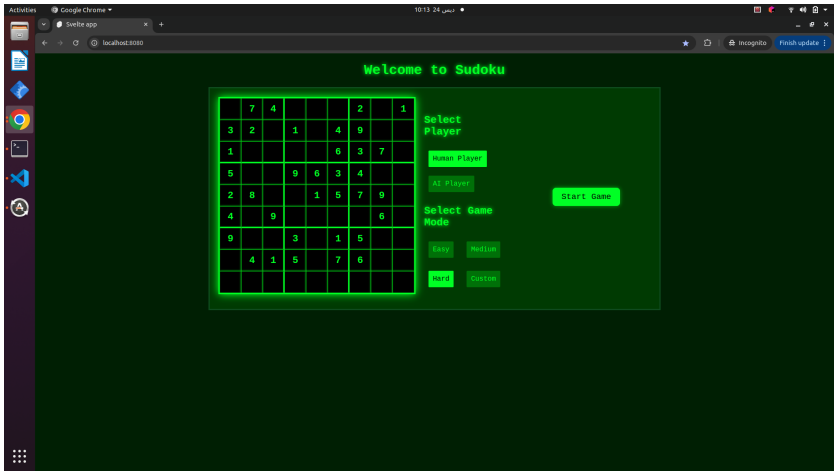

Figure 4: Initial Sudoku Board Hard

- AC-3 is effective for early pruning but adds computational overhead.
- Board difficulty directly impacts solving time and number of recursive calls.

8 Challenges

- Ensuring unique solutions in the generator.
- Optimizing AC-3 for larger grids.
- Handling edge cases like unsolvable puzzles.

9 Other Features

- Interactive Sudoku solver with GUI.
- Logging of constraint propagation and domain pruning.

10 Conclusion

This project implements efficient algorithms for solving and generating Sudoku puzzles, showcasing the trade-offs between different optimization techniques. Future work could explore hybrid approaches and advanced heuristics for further improvements.

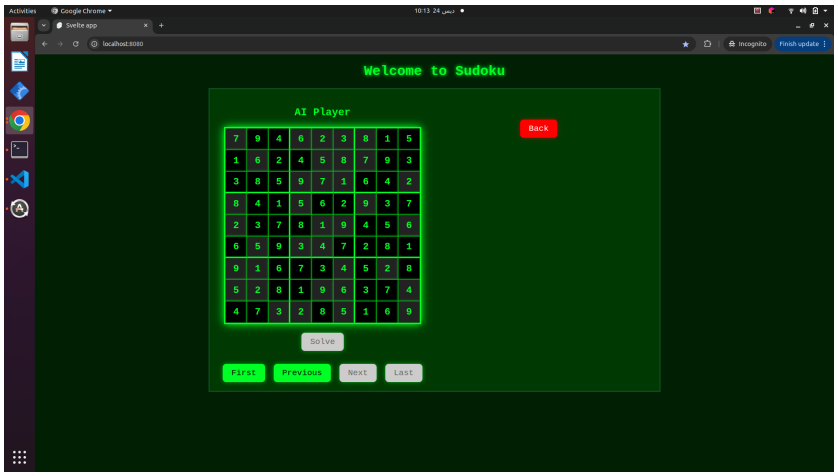

Figure 5: Solved Sudoku Board

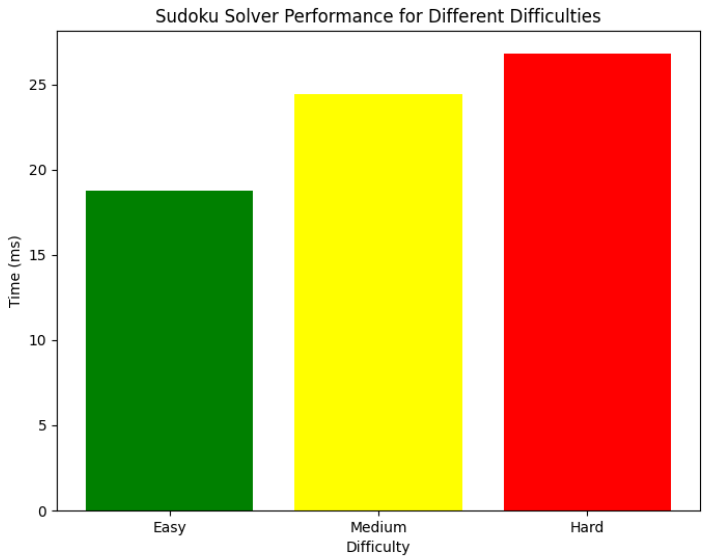

Figure 6: Solved Sudoku Board