AI Sudoku Solver Project Proposal

Joann Sum
CPSC 481

Problem Description
I am building a Sudoku solver with a difficulty estimation system. Unlike most solvers that just find a solution, mine will analyze and predict how challenging a puzzle would be for humans to solve. Using heuristic evaluation and search techniques from AI, the system will quantify difficulty based on the required solving techniques, the number of logical deductions needed, and the complexity of the solution path. This creates a more meaningful metric than just "how long the computer takes to solve it," making it useful for puzzle creators and players alike.

Programming Language

- JavaScript/TypeScript with Next.js for the frontend.
- Python backend for the solver algorithm if needed.

Datasets
No external datasets should be needed. I can hardcode some test puzzles.

Existing Code
I plan to build a Next.js app from where I will implement the core solving algorithms myself but might use some UI libraries to save time.

Algorithm/Approach
I will implement Sudoku as a constraint satisfaction problem using backtracking search with forward checking. My solver will use the Minimum Remaining Values heuristic to always select the most constrained cells first. I'll also implement the Least Constraining Value principle to maintain maximum flexibility in the remaining grid. For better performance, I might add Arc Consistency algorithms to propagate constraints throughout the puzzle before and during the search process. The novel aspect of my project will be the difficulty estimation system. This will analyze solving paths, count required advanced techniques, measure the average branching factor during solution, and calculate information entropy of the grid at various stages. These metrics will feed into a weighted evaluation function that can classify puzzles by difficulty level. For testing, I can compare the performance of different heuristic combinations across various puzzle difficulties to demonstrate the efficiency gains from proper AI technique selection.

Timeline

- Week 1 (Starting from April 21st):
    - Days 1-2: Set up Next.js project structure
    - Days 3-5: Implement core solver algorithm & Work on Presentation
    - Days 6-7: Create input/output UI
- Week 2:
    - Days 1-3: Testing and debugging, Optional features if time allows (step visualization, puzzle generation)
    - Days 4-6: Finalize project and presentation

Computing Platform
Standard laptop - no special hardware required.

Roles and Responsibilities
I have experience with web development from personal projects and algorithms from previous coursework which will be relevant to this project.