CS4300 Constraint Satisfaction Problem Assignment

Name: Connor Denney

GitHub Repository: https://github.com/Connor1031/ai-assignment-3-csp.git

1. Problem Domain: 4x4 Sudoku

The problem domain chosen for this assignment is a Sudoku puzzle, played on a 4x4 and 9x9 grid. The grid is divided into four 2x2 sub-grids and 9 3x3 sub-grids respectively.

The objective is to fill the 16 cells of the grid with digits from 1 to 4.

1. **Row Constraint:** Each digit (1, 2, 3, 4) must appear exactly once in each row.
2. **Column Constraint:** Each digit must appear exactly once in each column.
3. **Box Constraint:** Each digit must appear exactly once in each of the four 2x2 boxes.

The objective is to fill the 81 cells of the grid with digits from 1 to 9.

4. **Row Constraint:** Each digit (1, 2, 3, 4, etc) must appear exactly once in each row.
5. **Column Constraint:** Each digit must appear exactly once in each column.
6. **Box Constraint:** Each digit must appear exactly once in each of the 9 3x3 boxes.

A puzzle begins with some cells pre-filled with digits. The solver's task is to determine the correct digit for every empty cell while respecting all three constraints.

2. Formalization as <X, D, C>

The 4x4 Sudoku puzzle can be formally defined as a Constraint Satisfaction Problem (CSP) with the following components:

- **X (Variables):** A set of 16 variables, where each variable represents the cell in row *i* and column *j* (for *i*, *j* in {1, 2, 3, 4}). X={C11,C12,C13,C14,C21,...,C44}
- **D (Domains):** The domain for each variable is the set of possible digits that can be placed in a cell. For every variable Cij in X, the domain is: Dij={1,2,3,4}
- **C (Constraints):** A set of 12 alldiff constraints that enforce the rules of the game:
    - **4 Row Constraints:** alldiff(C11, C12, C13, C14), alldiff(C21, C22, C23, C24), etc.
    - **4 Column Constraints:** alldiff(C11, C21, C31, C41), alldiff(C12, C22, C32, C42), etc.
    - **4 Box Constraints:** alldiff(C11, C12, C21, C22), alldiff(C13, C14, C23, C24), etc.

The same goes for the 9x9 sudoku puzzle but with more variables respectively, domain up to 9, and the same constraints but with 27 of them.

3. Heuristic Implementation: Minimum Remaining Values (MRV)

To improve the efficiency of the provided backtracking solver, the **Minimum Remaining Values (MRV)** heuristic was used. This is a variable-ordering heuristic that dynamically selects which variable to assign next. Instead of processing variables in a fixed, arbitrary order, the MRV directs the solver to choose the unassigned variable that has the **fewest** legal values remaining in its domain. The rationale is to fail fast by tackling the most constrained parts of the problem first, the solver can quickly cut large sections of the search tree that would lead to a contradiction. This often reduces the backtracks required to find a solution.

The implementation involved creating a new solver function, solve_backtracking_mrv, which, at each step of the recursion, iterates through the unassigned variables to find the one with the smallest current domain size before attempting assignments.

4. Results and Reflection

The performance of the baseline solver was compared against the new MRV-enhanced solver on three Sudoku instances of increasing difficulty. The runtime to find all solutions was measured for each test.

Puzzle	Solver runtime	MRV Solver runtime
4x4 easy	.0005 s	.0005 s
4x4 medium	.0005 s	.0005 s
4x4 hard	.0009 s	.001 s
9x9 easy	.0357 s	.0252 s
9x9 medium	10.2816 s	.6273 s
9x9 hard	21.0929s	20.2137 s

As the results clearly show, the MRV heuristic provided a performance improvement across all the larger instances although for the 9x9 hard but I think that was just unlucky searching by the solver. The baseline solver's performance slows significantly as the number of initial clues

decreases and <X,D,C> increases, whereas the MRV solver's runtime remains consistently lower. This confirms the heuristic's effectiveness. By better navigating the search space, the MRV solver avoids exploring many fruitless branches, leading to a much more efficient search for a solution.