Backtracking with MRV Heuristic: A Synergistic Approach for Sudoku CSP Solving

Lam Chun Lung Leo

1. Statement of approach

Sudoku - a classic Constraint Satisfaction Problem (CSP) where variables are the 81 cells in the 9×9 grid, and the domains are numbers 1-9 for empty cells with constraints of row, column, and 3×3 box uniqueness. While the naive backtracking method serves as a standard approach to the problem, the algorithm complexity hinders it from practical application (1). However, the Minimum Remaining Values (MRV) heuristic demonstrates remarkable performance on backtracking methodology by harnessing a "fail-first" strategy to prune the search tree, improving overall computational complexity (2). Hence, to enhance the computational performance and maintainability of the script, this implementation employs the MRV heuristic with OOP structure which stores the grid state and main solving flow within the class.

2. Core algorithm description

For the main algorithm flow, the implementation starts by initialising the class function with an input puzzle, followed by the validation of the solvability. The algorithm then examines all the coordinates in the current grid to find the unassigned variables. When a candidate is found, the function computes the possible domains in the position by elimination of domains that conflict with the constraints and returns a list of the remaining. The algorithm continues to compare all unassigned variables and returns the one with the fewest possible domains when the full grid has been examined.

For the backtracking part, the search begins by the coordinate of the grid and the possible correlated domains from the heuristic output, followed by iteration of all the candidates. For each iteration, the loop starts by assigning the first domains in the list and calls the main solving function recursively to validate the assigned input, constructing a deep-first search tree. Once the bottom is reached, the backtracking function propagates back on to the success of assigning all variables. In case no valid domain values remain for a variable, the algorithm resets the state to unassigned again and moves on to the next possible domains in the previous depth, achieving backtracking.

3. Optimisation

In terms of the optimisation of the search method, instead of brutal force search for all cells, MRV Heuristic approach prioritises the cells with fewest possible domains and leverages forward checking, enabling early detection of dead-end paths. While the time complexity formula remains unchanged, the reduction of the branching factor reduces the tree size exponentially, avoiding the expedition on invalid nodes.

$$O(b^d) \rightarrow O(b'^d), \quad \text{where} \quad b' < b$$

Additionally, the initial validation of input arrays filters out unsolvable and invalid puzzles, enhancing the overall robustness and preventing the waste of computation force on malformed puzzles. The quantitative metric across all levels of puzzle with mean and maximum computation time has been provided as below:

Table 1. Quantitative Metrics and Timing

Metric	Very Easy	Easy	Medium	Hard
Mean Time (ms)	1.140	0.836	1.050	322.1
Max Time (ms)	2.001	1.003	1.260	1713.14

4. Reflections and suggestions for further work

From the strength perspective, the implementation efficiently tackles the sudoku puzzles by MRV approach, resulting in average resolving time around 1 ms for most of the easy-medium questions and perfect correctness for all cases. The application of clean OOP structure contributes to maintainability and extensibility. However, the exponential nature of MRV's complexity hinders the performance of CSP resolution of puzzles with minimal clues, causing a considerable escalation in time taken for hard instances. For example, the computation time of question 15 in hard sudoku exceeded 1 second. The lack of degree heuristic leads to the failure of identifying future constraints caused by specific cell selection.

Although the worst case scenario remains a challenge, the literature proposes an alternative method - Dancing Link (DLX) which leverages sparse binary matrices and Donald Knuth's Algorithm X to further reduce the complexity of sudoku puzzles (3). Compared to the MRV method, DLX approach offers a better per-step computational overhead by removal and restoration of constraints, achieving enhanced performance on validating available domains for variables. Hence, the implementation of DLX methodology serves as a feasible approach for further work.

References

[1] F. Bacchus and P. van Run. Dynamic variable ordering in CSPs. In *Principles and Practice of Constraint Programming (CP-95)*, pages 258–275, 1995. Available at: http://www.cs.toronto.edu/~fbacchus/Papers/BvRCP95.pdf

[2] S. Russell and P. Norvig. Constraint satisfaction problems. In *Artificial Intelligence: A Modern Approach*, chapter 5, 4th edition, Pearson, 2021. Available at: http://aima.cs.berkeley.edu/newchap05.pdf

[3] H. Læstander and M. Harrysson. Solving Constraint Satisfaction Problems using Backtracking and Local Search, 2014. Available at: https://www.kth.se/social/files/58861771f276547fe1dbf8d1/HLaestanderMHarrysson_dkand14.pdf