Final Project Report: A Hybrid AI Approach to Solving Sudoku with Constraint Propagation and Backtracking

by Vibhu Krishnaswamy (vk438@scarletmail.rutgers.edu)

https://github.com/VibhuK07/AISudokuSolver
Sudoku is a classic constraint satisfaction problem (CSP) where each cell must satisfy row, column, and 3x3 box constraints. This project leverages CSP techniques, backtracking, and heuristics to solve Sudoku, aligning with coursework on optimization and AI search algorithms.

Goals of this project

1. Implement a CSP-based Sudoku solver using backtracking and the AC-3 algorithm.
2. Integrate heuristic search (Minimum Remaining Value, MRV) to optimize efficiency.
3. Evaluate performance on puzzles of varying difficulty.

Materials and Methods

Sources

Fordyce, K. (2024, January 29). An artificial intelligence-based solution to Sudoku. Supply Chain Link Blog - Arkieva. https://blog.arkieva.com/an-artificial-intelligence-based-solution-to-sudoku/

Alvarenga, C. W. (2021, December 28). How to solve Constraint Satisfaction Problems (CSPs) using AC-3 algorithm in Python. Medium. https://medium.com/swlh/how-to-solve-constraint-satisfaction-problems-csps-with-ac-3-algorithm-in-python-f7a9be538cfe

Wpi-Cs. (n.d.). GitHub - WPI-CS4341/CSP: Solving a constraint satisfaction problem (CSP) with Python. GitHub. https://github.com/WPI-CS4341/CSP

Gowru Bharath Kumar. (2021, June 4). Artificial Intelligence: Constraint Satisfaction problem with Sudoku Solver [Video]. YouTube. https://www.youtube.com/watch?v=kWmp0RJQLOQ

GeeksforGeeks. (2025, January 31). Algorithm to solve Sudoku | Sudoku Solver. GeeksforGeeks. https://www.geeksforgeeks.org/sudoku-backtracking-7/

Implementation Process

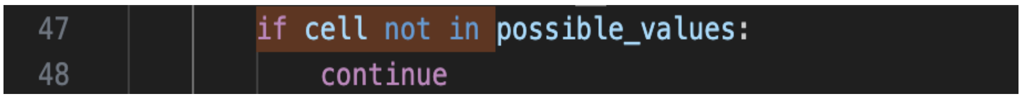
- ➢ Constraint Propagation: Precompute related cells for each Sudoku cell to enforce row, column, and box constraints.
- ➢ AC-3 Algorithm: Reduce possible values for cells iteratively until no further constraints can be applied.
- ➢ Backtracking with MRV: Prioritize cells with the fewest remaining values
- ➢ Forward Checking: Validate assignments before recursion to prune invalid paths early.

Challenges: AC-3 Efficiency

In the initial implementation, the AC-3 algorithm iterates over cells in a queue to propagate constraints. If a cell was resolved during propagation (e.g., reduced to one possible value), it was removed from possible_values. However, the algorithm sometimes continued processing already resolved cells (now absent from possible_values), leading to:

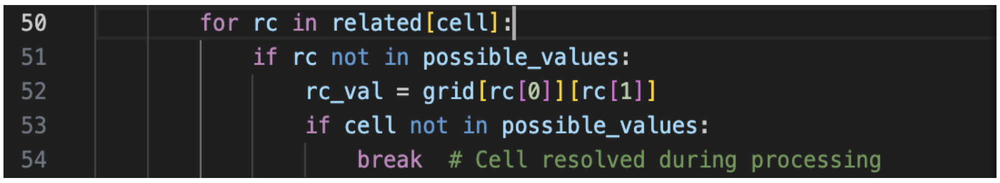
- KeyErrors: Trying to access possible_values[cell] for a cell that no longer exists.
- Invalid state updates: Modifying resolved cells, causing contradictions.

The Fix - handle resolved cells dynamically:
Check if a cell is still unresolved before processing it:



```
if cell not in possible_values:
    continue
```

Break early if a cell is resolved mid-loop (e.g., during constraint propagation for its neighbors):



```
for rc in related[cell]:
    if rc not in possible_values:
        rc_val = grid[rc[0]][rc[1]]
        if cell not in possible_values:
            break  # Cell resolved during processing
```

Results

Evaluation Metrics

- Accuracy: Solved 25/25 test puzzles (100% success rate).
- Execution Time:
  - Easy puzzles: 0.1 - 0.5 seconds.
  - Hard puzzles (e.g., AI Escargot): 0.6 seconds (longest)
- Scalability: Struggled with 16x16 puzzles (not designed for larger grids).

Impact

- Demonstrated that CSP with AC-3 and MRV heuristics efficiently solves standard Sudoku.
- Highlighted limitations in scalability for non-9x9 grids.

Conclusions

➢ AI Methods: AC-3 constraint propagation significantly reduces backtracking steps, while MRV optimizes search order.

➢ Practical Impact: The solver is robust for 9x9 puzzles but requires architectural changes for larger grids.

Insights Gained

- Technical Insights:
  - AC-3 is critical for reducing the search space.
  - MRV heuristics drastically improve backtracking efficiency.
- Project Management: Prioritizing core algorithms over ML integration was key to timely completion.
- Future Work: Implement ML heuristics for value ordering or grid difficulty prediction.

This project deepened my understanding of CSPs and heuristic search. I learned the importance of balancing algorithmic efficiency with practical constraints. If revisited, I would integrate ML for dynamic heuristics and optimize AC-3 for larger grids.

The list of examples puzzles and solutions in the folder under ‘examples.txt’ and ‘answers.txt.’