Rashedul Islam Tono | Wahidun Akter | Nusrat Zahan
CSE440 | Section 01 | Group 03

Logic Puzzle Solver: Sudoku with AI

I.Introduction

Sudoku is a popular logic-based puzzle game played on a 9×9 grid, where the goal is to fill the grid with numbers from 1 to 9. The challenge lies in ensuring that every row, column, and 3×3 sub-grid contains each number only once. This project focuses on implementing a Sudoku solver using Constraint Satisfaction Problems (CSP) and backtracking algorithms. The CSP approach reduces the solution space by applying constraints iteratively, simplifying the puzzle before applying backtracking. Backtracking ensures a systematic exploration of possibilities when constraints alone cannot solve the puzzle. To enhance the user experience, the solver is integrated with Pygame also Gradio interface, offering an interactive platform for Sudoku puzzles and it's solutions. The report discusses the methodology, implementation challenges, and performance analysis.

II. METHODOLOGY

Game Logic:

The game logic for Sudoku involves managing a 9×9 grid, validating inputs with is_valid, and solving puzzles using constraint_propagation and backtracking algorithms.

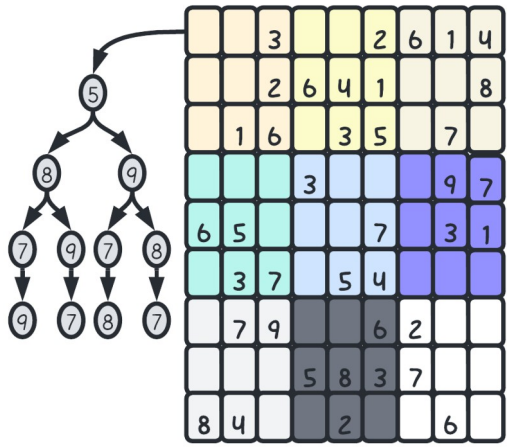


Figure: Game Tree

Functions like find_empty locate empty cells, while handle_input updates the board after validation. Players can reset puzzles using restart_game, ensuring smooth game-play, and the ability to solve effectively.

Backtracking Algorithm:

The backtracking algorithm solves Sudoku by systematically exploring possible values for each empty cell. It starts by identifying the first empty cell and tries numbers from 1 to 9. For each number, it checks if placing it adheres to Sudoku rules (unique in its row, column, and 3×3 grid). If valid, the number is placed, and the algorithm recursively attempts to solve the remaining puzzle. If no valid number works, it backtracks by removing the last placed number and tries the next option. This process repeats until the puzzle is solved or no solution exists, ensuring a systematic and exhaustive exploration of possibilities.

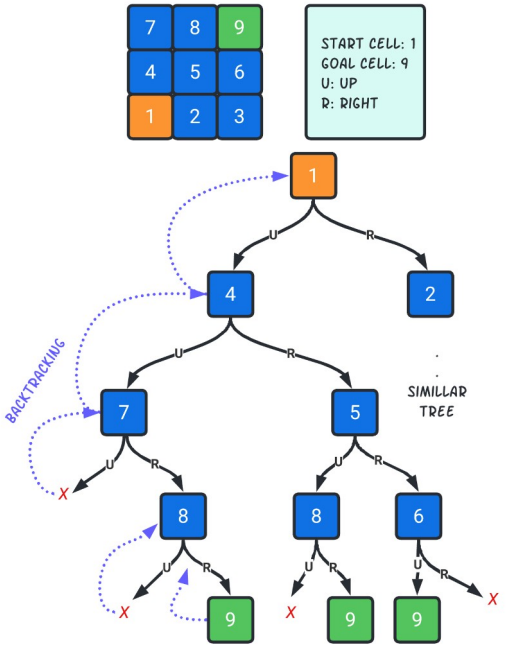


Figure: Backtracking Algorithm

Constraint Satisfaction Problems (CSP)

Sudoku involve defining variables (81 cells), domains (possible values 1-9), and constraints (no duplicates in rows, columns, and 3×3 sub-grids). Constraint propagation simplifies puzzles by reducing domains based on existing values. Backtracking explores assignments for unsolved cells, ensuring all constraints are met. This combination efficiently solves Sudoku by pruning invalid possibilities early and systematically handling more complex scenarios when propagation alone isn't sufficient.

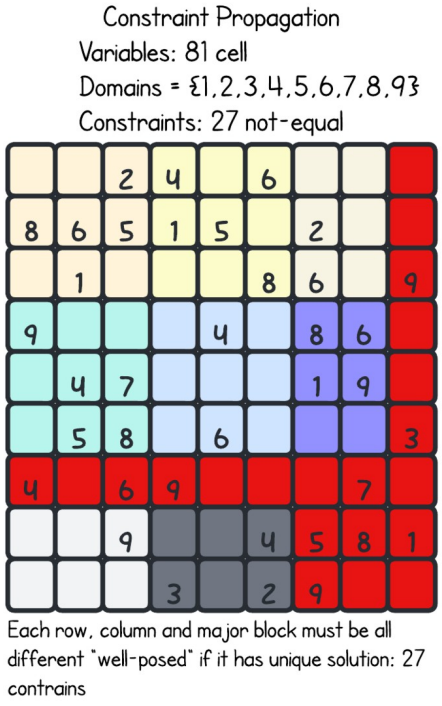


Figure: Constraint Propagation

III. User Interface

Pygame interface

The Pygame interface creates a visually rich and interactive environment for users who enjoy solving puzzles locally. It features a 9×9 grid where players can click on any cell and input numbers directly from their keyboard. To make the process smoother, the system provides real-time feedback by validating entries instantly. For example, it highlights errors or conflicts, helping users follow Sudoku's rules without frustration. A unique and fascinating aspect of this interface is its ability to visually demonstrate how the backtracking algorithm works.

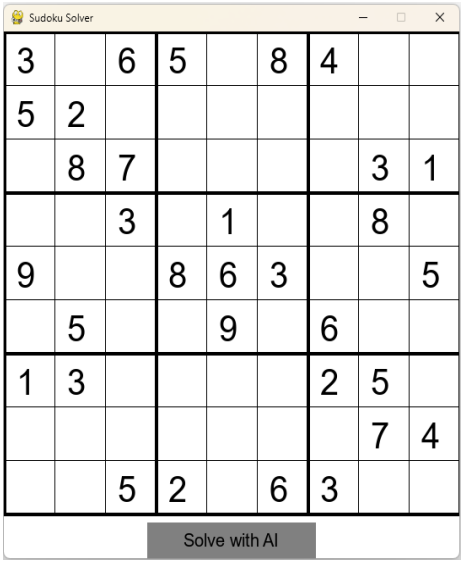


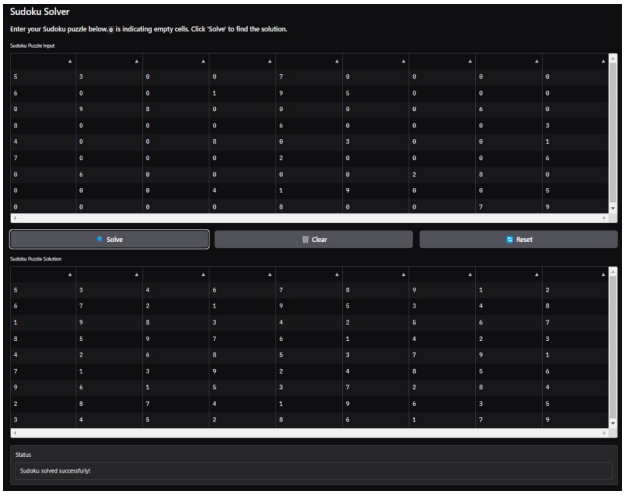
Figure: Pyboard

As the algorithm solves the puzzle, users can watch it fill in the cells step by step, making the logic behind the solution easy to understand and engaging to follow.

Gradio Interface

The Gradio interface extends the solver's reach beyond local environments by providing a simple and accessible web-based solution. Users can upload their Sudoku puzzles as images or text files and receive completed solutions almost instantly. The interface is designed to work seamlessly on any device, from desktops to mobile phones, ensuring that users can access it wherever they are. This makes it especially useful for those looking for quick and hassle-free puzzle solving without the need for additional software.

(https://huggingface.co/spaces/tono-rashedul/Sudoku-Solver).



IV. Results

The project successfully implemented a fully functional Sudoku solver with advanced techniques. Using constraint propagation and backtracking, the solver efficiently handles puzzles of varying difficulty. Constraint propagation reduces possibilities by enforcing rules, while backtracking ensures accurate solutions for complex cases. The program is robust, correctly solving scenarios from simple to challenging puzzles.

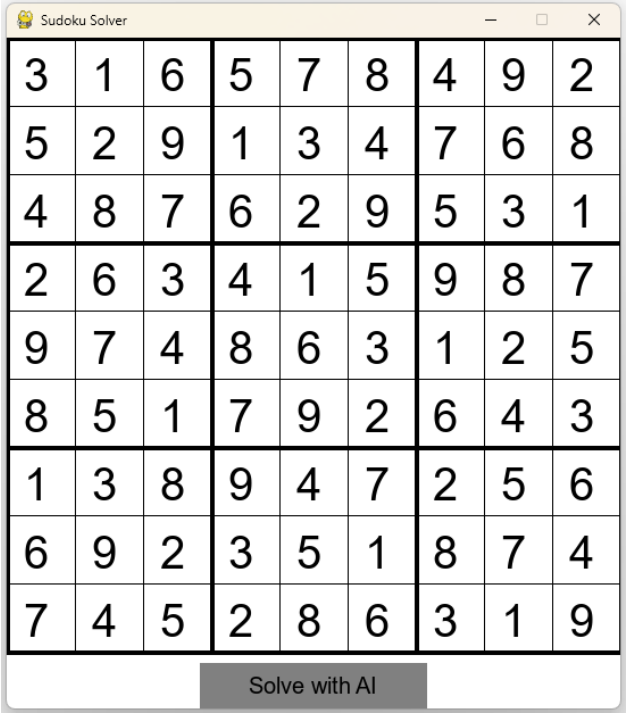


Figure: Pyboard (Solve With AI)

Additionally, a pygame interface allows easy input of puzzles and displays solutions clearly, enhancing usability and practical application.

V. Technical Challenges

Since the project was assigned before the chapters related to the topic were covered, we initially faced several challenges in implementing the Sudoku solver. The major difficulties encountered are as follows:

- **Understanding Constraint Propagation:** Initially, learning how to propagate constraints effectively to reduce possible domains for each cell was challenging. However, with online tutorials and examples, we gained clarity and successfully applied this technique.
- **Implementing Backtracking Efficiently:** Designing the recursive backtracking algorithm posed difficulties, particularly in managing the assignment and rollback of values. Through iterative testing, we resolved issues where conflicts were not correctly undone.
- **Row, Column, and Box Constraints:** Ensuring that values did not repeat in rows, columns, or sub-grids was complex at first. Debugging helped us refine the validation function to accurately enforce these constraints.
- **Handling Unsolvable Puzzles:** Detecting and handling scenarios where puzzles had no solution was initially missed. We addressed this by adding checks for invalid states during backtracking.
- **Performance Optimization:** For harder puzzles, the solver was slow. By combining constraint propagation with backtracking and using early termination

for invalid states, we improved performance significantly.

These challenges provided valuable learning experiences and enhanced our problem-solving skills.

VI. Conclusion

This project successfully demonstrates the application of constraint propagation and backtracking to develop an efficient Sudoku solver. The integration of a user-friendly interface further enhances usability, enabling seamless input of puzzles and display of solutions. The combination of these techniques ensures accurate and reliable solving for puzzles of varying difficulty.

VII. Acknowledgment

We would like to thank our respected faculty **Mohammad Shifat-E-Rabbi**, for his constant support. Also heartfelt gratitude to the users who interacted with the game and helped us to find the strategies.

VIII. References

1. GeeksforGeeks, “Sudoku Backtracking Algorithm,” [Online]. Available: https://www.geeksforgeeks.org/sudoku-backtracking-7/.

2. Medium, “Recursive Backtracking for Sudoku Solver,” [Online]. Available: https://medium.com/swlh/recursive-backtracking-for-combinatorial-path-finding-and-sudoku-solver-algorithms-1bf46ba2838c.

3. IbanCG, “Sudoku Solver,” [Online]. Available: https://ibancg.github.io/Sudoku-Solver/.

4. GeeksforGeeks, “Constraint Satisfaction Problems (CSP) in Artificial Intelligence,” [Online]. Available: https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/.

5. Medium, “Formulation of CSP Problem: Sudoku Puzzle,” [Online]. Available: https://medium.com/@co.2020.prkude/formulation-of-csp-problem-sudoku-puzzle-7d5e1d547382.

6. Wikipedia, “Sudoku,” [Online]. Available: https://en.wikipedia.org/wiki/Sudoku.

7. Iyer R, Jhaveri A, Parab K. A Review of Sudoku Solving using Patterns [J]. International Journal of Scientific and Research Publications, 2013.