Sanjivani College of Engineering, Kopargaon

(An Autonomous Institute)

Department Of Computer Engineering

(T.Y. Computer)

2024-2025

"Sudoku Solver"
Creating a Sudoku Solver Using Backtracking

Prof. P. B. Dhanwate

[Guide]

Dr. D. B. Kshirsagar

[H.O.D. Comp Engg]

Dr. A. G. THAKUR

[Director]

CIA Activity: Sudoku Solver Using Backtracking Algorithm

Abstract

This paper presents the implementation of a Sudoku solver using the backtracking algorithm. Sudoku is considered a constraint satisfaction problem where the goal is to fill a 9×9 grid such that each row, column, and subgrid contains digits from 1 to 9 exactly once. This study provides a step-by-step breakdown of the algorithm, explores its efficiency, and addresses edge cases to ensure optimal performance.

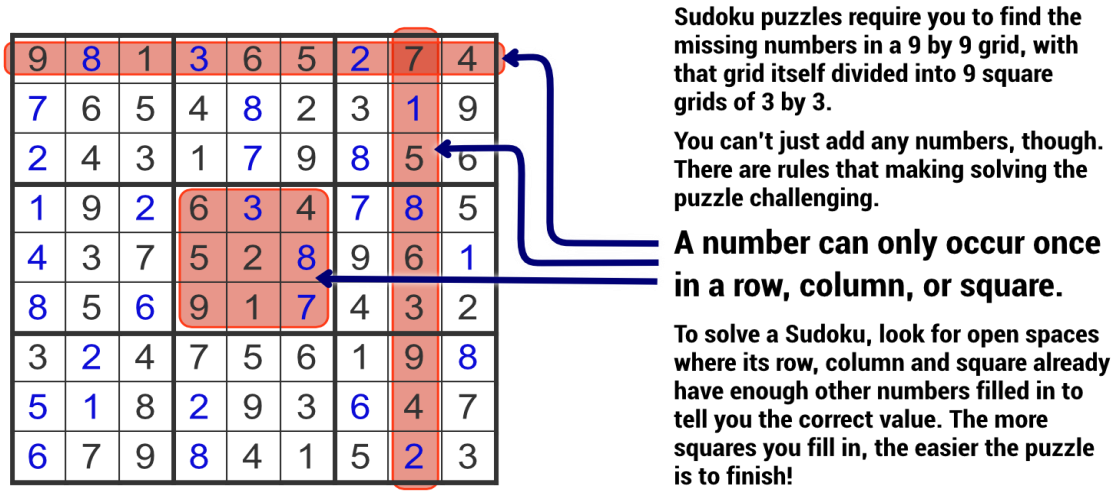
Keywords

Sudoku, Backtracking Algorithm, Constraint Satisfaction Problem (CSP), Sudoku Solver, Algorithm Optimization

1. Introduction

Sudoku is a popular puzzle game that requires players to fill a 9×9 grid such that each row, column, and 3×3 subgrid contains all digits from 1 to 9 without repetition. Solving Sudoku can be framed as a constraint satisfaction problem (CSP), making it an interesting challenge

for algorithm design in computer science. This paper focuses on implementing a Sudoku solver using the backtracking algorithm.



2. Methodology

The input to the problem is a partially filled 9×9 grid, where each empty cell is represented by 0. The objective is to complete the grid such that every row, column, and 3×3 subgrid contains the digits from 1 to 9 exactly once. The backtracking algorithm is an effective approach to solve this, by attempting to fill each cell with a valid number, checking constraints, and backtracking if constraints are violated.

- The backtracking algorithm proceeds with the following steps:

1. Represent the Sudoku grid as a 9×9 matrix.

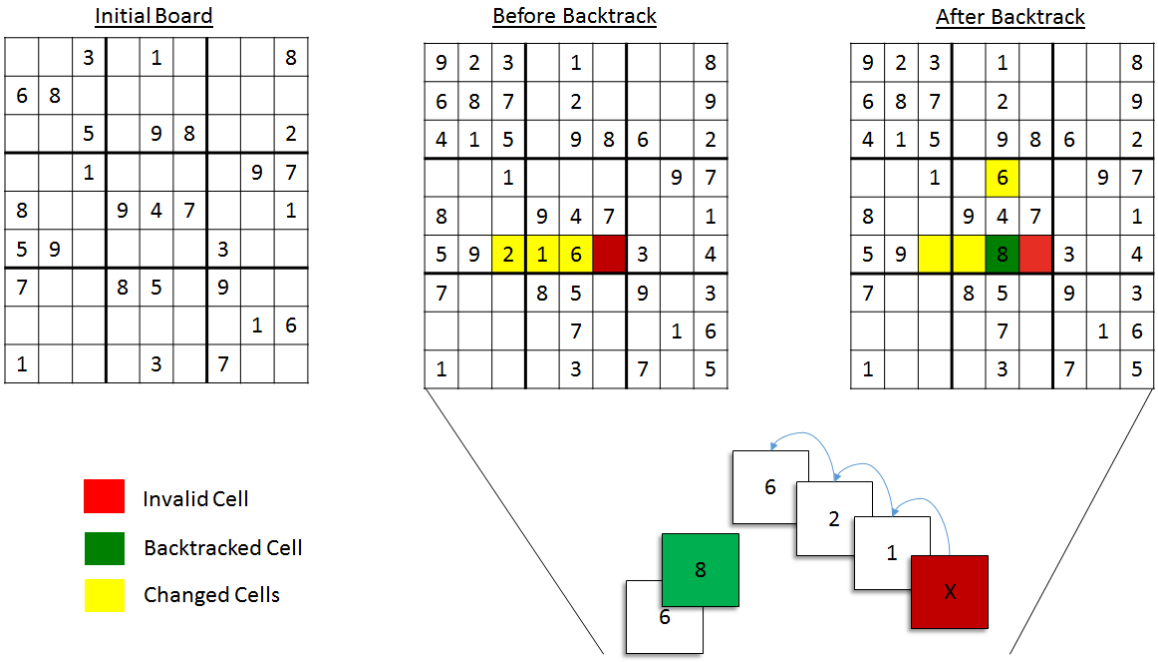
2. For each empty cell, attempt to place digits 1 through 9.

3. Check constraints: the number should not appear in the same row, column, or subgrid.

4. If a valid number is placed, recursively attempt to solve the next empty cell.

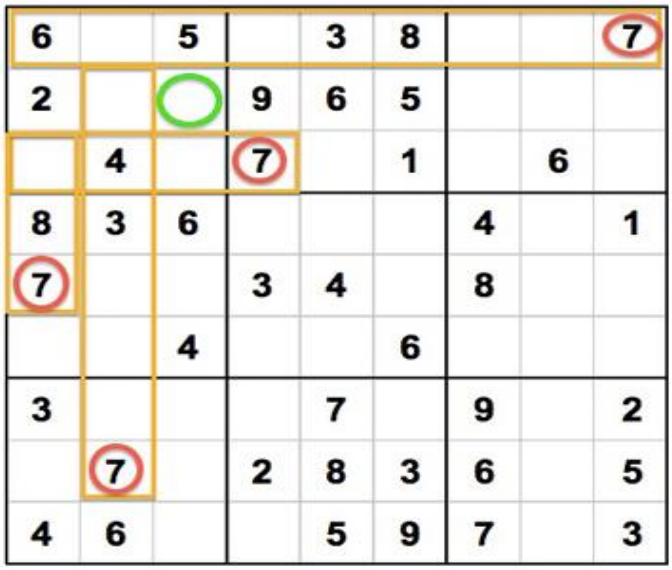
5. If no valid number can be placed, backtrack by removing the number and trying a different one.

6. The process repeats until the grid is fully solved or it is determined that the puzzle is unsolvable.



3. Problem Explanation and Requirement Analysis

Sudoku is a constraint-based puzzle that requires attention to detail and adherence to specific rules. The backtracking algorithm is ideal for solving the problem by testing possible digits in a recursive manner. This section describes the problem and the specific requirements for the Sudoku solver, such as constraint checking for rows, columns and subgrids.



4. Results and Discussion

The implementation of the backtracking algorithm successfully solves a wide range of Sudoku puzzles, handling edge cases such as minimal clues. However, performance can vary based on the complexity of the puzzle and the number of empty cells. Future work could focus on optimizing the algorithm further using techniques such as constraint propagation or more advanced heuristics.

5. Conclusion

This paper presented the design and implementation of a Sudoku solver using the backtracking algorithm. The algorithm effectively fills the grid by recursively testing possible numbers in empty cells while adhering to the constraints of the puzzle. Further optimizations could be explored to enhance its performance for more complex Sudoku puzzles.

References

[1] A. Author, 'Title of the Paper', Journal Name, vol. 10, no. 2, pp. 150-160, 2023.

[2] B. Author, 'Sudoku and Backtracking', Conference on Algorithmic Puzzles, 2022.