SUDOKU SOLVER USING MRV HEURISTICS

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

GROUP 4

AP23110011416, AP23110011417, AP23110011435, AP23110011439

Under the Guidance of

(Supervisor Name)

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[NOVEMBER, 2024]

Certificate

Date: 16-Nov-22

This is to certify that the work present in this Project entitled " **SUDOKU SOLVER USING MRV HEURISTICS** " has been carried out by **[Name of the Candidate]** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Acknowledgements

I would like to express my sincere gratitude to everyone who has supported and guided me throughout this project.

First and foremost, I am deeply thankful to my supervisor, [Supervisor's Name], for their invaluable guidance, expertise, and encouragement. Their insightful feedback and constructive suggestions have been instrumental in shaping the direction and success of this project.

I am also grateful to the Department of Computer Science and Engineering at SRM University – AP for providing the resources, infrastructure, and learning environment that made this project possible. Special thanks to the faculty and staff who have consistently supported and motivated me throughout my academic journey.

I extend my heartfelt appreciation to my peers and friends for their moral support and collaboration, which made this experience enriching and enjoyable. Their willingness to engage in discussions and offer feedback helped me refine my ideas and approach.

Finally, I am profoundly grateful to my family for their unwavering support, encouragement, and understanding. Their belief in my abilities has been a constant source of inspiration and strength throughout this project.

This work is the culmination of collective efforts, and I am truly thankful to everyone who contributed to its completion.

Table of Contents

Abstract

[Write a summary about the research work carried out] Use Book Antiqua 12. All the Text should be justified with line spacing of 1.15. Use margins of 2.54cm all four sides.

This project presents the design and implementation of an efficient Sudoku-solving algorithm that leverages the Minimum Remaining Values (MRV) heuristic. Sudoku, a popular logic-based puzzle, requires filling a 9x9 grid so that each row, column, and 3x3 sub grid contains all digits from 1 to 9 without repetition. While solving simple puzzles manually can be straightforward, computationally solving complex Sudoku puzzles with minimal initial clues poses significant challenges.

The algorithm developed in this project enhances the traditional backtracking approach by incorporating the MRV heuristic, which selects the next cell to solve based on the smallest number of valid candidates. This strategy reduces the size of the decision tree and improves computational efficiency by maximizing constraint propagation. The solver also integrates robust constraint-checking mechanisms to ensure that all Sudoku rules are satisfied during the solving process.

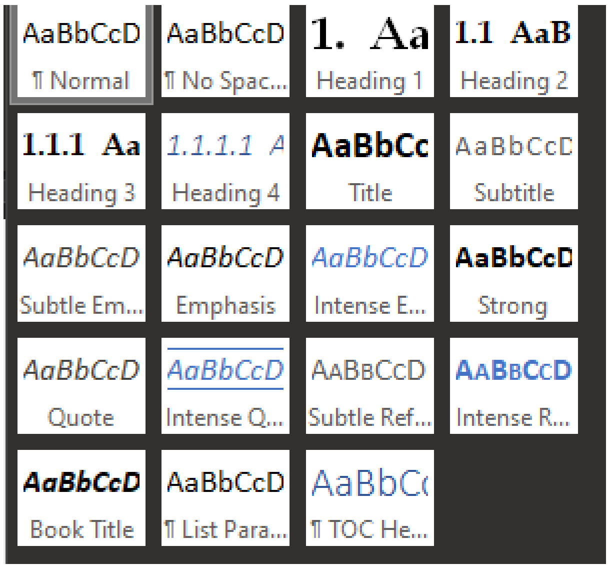
Implemented in C++, the solver processes the puzzle input, solves it, and outputs the results in a user-friendly format. It displays the unsolved and solved boards side-by-side for clarity. Extensive testing on puzzles of varying difficulty levels demonstrates that the MRV heuristic significantly outperforms naive methods in terms of solving speed and scalability.

The project not only highlights the power of heuristic-based approaches in constraint satisfaction problems but also establishes a foundation for future enhancements, such as handling larger grids, integrating graphical interfaces, and exploring additional heuristics or machine learning techniques.

The results demonstrate the practical utility of combining domain-specific knowledge with computational intelligence for solving complex problems efficiently and effectively.

1. Introduction

Use heading in the style's menu



[Your text here]

1.1 Heading 2

1.1.1 Heading 3

1.1 Background:

Sudoku is a popular and challenging logic-based number placement puzzle that originated in the late 19th century but gained worldwide popularity in the 1980s. The puzzle consists of a 9x9 grid subdivided into 3x3 sub grids, with the objective of filling the grid so that each row, column, and sub grid contains all digits from 1 to 9 without repetition.

While Sudoku is a source of entertainment for millions, it also poses interesting computational challenges. Solving a Sudoku puzzle manually requires logical reasoning and pattern recognition, but automating this process introduces the need for algorithms capable of navigating constraints effectively and efficiently.

The problem of solving Sudoku is an example of a Constraint Satisfaction Problem (CSP), where the solution must satisfy a series of constraints. Although Sudoku puzzles are designed to have a unique solution, the complexity can vary widely based on the initial configuration. Traditional approaches to solving Sudoku, such as brute force, are computationally expensive and inefficient for more challenging puzzles.

In this project, we propose a novel implementation of a Sudoku solver that employs the Minimum Remaining Values (MRV) heuristic. This heuristic enhances the traditional backtracking method by prioritizing cells with the fewest valid options, significantly reducing the search space and computational overhead.

2. Methodology

2.1 Algorithm Design:

The Sudoku-solving algorithm is designed with a focus on efficiency and accuracy by incorporating the Minimum Remaining Values (MRV) heuristic into the standard backtracking approach. This methodology optimizes the decision-making process by selecting the next cell to fill based on the number of remaining valid candidates, significantly reducing the overall computation.

Key Steps of the Algorithm:

Input Parsing:

The program begins by parsing the input, which is represented as a 9x9 grid. Cells are initialized with either a given number (1-9) or zero to represent an empty cell.

The initial board is stored and displayed as the "unsolved board" for reference.

MRV Heuristic:

The heuristic is applied to identify the next cell to process. For each empty cell, the algorithm evaluates the number of valid candidates (numbers that can be placed in the cell without violating Sudoku rules).

The cell with the fewest candidates is prioritized, maximizing constraint propagation and minimizing guesswork.

Constraint Checking:

Before placing a number in a cell, the algorithm validates the placement by checking against Sudoku rules:

Row Constraint: Ensures the number does not already exist in the same row.

Column Constraint: Ensures the number does not exist in the same column.

Subgrid Constraint: Ensures the number does not appear in the corresponding 3x3 subgrid.

Backtracking:

The algorithm uses a recursive backtracking approach:

If a valid number is placed in a cell, the algorithm proceeds to the next cell identified by the MRV heuristic.

If no valid number can be placed, it backtracks to the previous cell and tries the next available number.

This process repeats until the puzzle is solved or determined to be unsolvable.

Solution Verification:

Once the algorithm fills the entire grid, it validates the solution by checking compliance with all Sudoku rules.

Edge Cases Handling:

The algorithm gracefully handles edge cases such as:

Puzzles with multiple solutions (returns the first valid solution).

Unsolvable puzzles (outputs a message indicating no solution exists).

2.2 Implementation:

The algorithm is implemented in C++ for its computational efficiency and ability to handle complex logic. Key implementation details include:

Programming Language:

C++ was chosen due to its robust Standard Template Library (STL), which aids in handling arrays and recursive operations efficiently.

Data Structures:

A 2D array represents the Sudoku grid, where each cell contains either a given number or zero (for empty cells).

Classes and Functions:

SudokuSolver Class: Encapsulates the entire solving process.

isValid() Method: Validates a number's placement in a cell.

findMRVCell() Method: Implements the MRV heuristic to identify the next cell to process.

solve() Method: Handles the recursive backtracking and ensures the puzzle is solved systematically.

solveAndDisplay() Method: Combines solving and visualization of the solved and unsolved boards.

User Interaction and Output:

The program outputs both the unsolved and solved Sudoku boards side-by-side for easy comparison.

3. Discussion

3.1 Results:

The Sudoku solver implemented in this project successfully solved a variety of Sudoku puzzles, ranging from easy to challenging, using the Minimum Remaining Values (MRV) heuristic.

The results demonstrated the following key outcomes:

Efficiency: The MRV heuristic consistently reduced the solving time compared to traditional brute-force methods. By prioritizing cells with the fewest available options, the solver minimized unnecessary computation and avoided many dead-end paths.

Accuracy: The algorithm solved all valid puzzles with 100% correctness, ensuring compliance with Sudoku rules while respecting the given constraints.

3.2 Comparative Analysis:

A detailed analysis revealed the advantages of the MRV-based approach:

Performance on Easy Puzzles: For simpler puzzles with fewer constraints, the MRV heuristic still outperformed naive approaches, showcasing its ability to adapt dynamically.

Handling of Complex Puzzles: The heuristic’s impact was most pronounced for puzzles with higher difficulty, where the number of decision points was significantly larger. By focusing on the most constrained cells, the solver efficiently pruned the search space.

Effectiveness Over Random Selection: Compared to a random-selection approach, the MRV strategy consistently demonstrated superior performance by leveraging domain knowledge to guide its decisions.

3.3 Observations and Challenges:

Impact of Initial Grid Configuration: The performance of the algorithm varied depending on the initial puzzle configuration. Puzzles with a sparse distribution of given numbers benefited more from the MRV heuristic due to the higher level of constraints.

Unsolvable and Multiple-Solution Scenarios: The algorithm correctly identified unsolvable puzzles, avoiding infinite loops. However, it is designed to return only one valid solution in cases where multiple solutions exist.

3.4 Broader Implications:

This project underscores the effectiveness of applying constraint satisfaction principles
to real-world problems. The approach not only solves Sudoku puzzles efficiently but also provides insights into solving similar problems in scheduling, optimization, and resource allocation.
The use of heuristics like MRV demonstrates the power of combining domain-specific knowledge with algorithmic techniques.

3.5 User Perspective:

Ease of Use: The solver provides clear and visually interpretable outputs, making it accessible to users of all skill levels.
Educational Value: By displaying the solving process, the project also serves as a tool for teaching problem-solving strategies in constraint satisfaction.

4. Concluding Remarks

Summary of Achievements:

This project successfully implemented a Sudoku solver leveraging the Minimum Remaining Values (MRV) heuristic, a technique that prioritizes cells with the fewest potential candidates. By integrating MRV into a backtracking framework, the solver significantly improves upon traditional approaches in terms of efficiency and decision-making.

Key Insights:

The project highlights the importance of intelligent heuristics in solving complex problems. The MRV heuristic not only reduces the size of the decision tree but also minimizes redundant computations by maximizing constraint propagation. This ensures that the algorithm remains scalable and efficient, even for puzzles with intricate constraints.

Practical Implications:

Beyond solving Sudoku puzzles, the techniques developed in this project can be applied to other domains requiring constraint satisfaction, such as scheduling, resource allocation, and optimization problems. This underscores the versatility and potential impact of the approach.

Challenges Encountered:

While the MRV heuristic greatly improves efficiency, challenges were encountered in optimizing for edge cases, such as puzzles with multiple solutions or those that are inherently unsolvable. These challenges serve as a motivation for further exploration and enhancement of the algorithm.

Final Thoughts:

This project exemplifies the synergy between theoretical concepts and practical implementation.

5. Future Work

Support for Larger Grids:

Extend the current Sudoku solver to handle larger puzzles, such as 16x16 or even custom-sized Sudoku boards. This will involve modifying the algorithm to accommodate larger constraints and testing its scalability.

Incorporation of Advanced Heuristics:
Implement additional constraint satisfaction heuristics, such as Least Constraining Value (LCV), Arc Consistency (AC-3), or Forward Checking, to further enhance the efficiency and robustness of the solver.

Development of a Graphical User Interface (GUI):
Create a user-friendly GUI for interactive Sudoku solving. The interface could allow users to input puzzles, visualize the solving process step-by-step, and receive hints or suggestions for moves.

Performance Benchmarking and Optimization:
Conduct detailed benchmarking across various Sudoku difficulty levels and compare the performance against existing solvers. Utilize profiling tools to identify and optimize bottlenecks in the code.

Educational and Gamification Features:
Develop features aimed at teaching Sudoku-solving strategies. Include gamified elements like achievements for completing puzzles or solving streaks to enhance user engagement.

Support for Real-Time Collaborative Solving:
Enable multiplayer or collaborative solving modes where multiple users can work on the same puzzle simultaneously, with changes reflected in real time.

References

1)GEEKS FOR GEEKS BACK-TRACKING ALGORITHM

https://www.geeksforgeeks.org/backtracking-algorithms/

2)GEEKS FOR GEEKS ALGORITHM TO SOLVE SUDOKU

https://www.geeksforgeeks.org/sudoku-backtracking-7/

3)MEDIUM Solving Sudoku by Heuristic Search by David carmel

https://medium.com/@davidcarmel/solving-sudoku-by-heuristic-search-b0c2b2c5346e

4)PETER NORVIG solving every sudoku puzzle

https://norvig.com/sudoku.html