Sudoku Vision Project (Foundations of AI: Multiagent Systems)

Jaidev Sanjay Khalane (22110103), Vannsh Jani (22110279)

I. Introduction and Aim

Sudoku is an interesting single-player game. Backtracking is a commonly used standard method to solve this puzzle. However, we can also optimize this algorithm by using some variable selection heuristics. Apart from this, we can also explore modelling sudoku puzzles as constraint satisfaction problems and solve them using some CSP solving algorithms like Forward Checking and AC3. In this project, we aim to explore solving sudoku puzzles using various algorithms as discussed above, such as Backtracking, Backtracking with heuristics, Forward Checking and AC3. Apart from that, we also aim to explore an optimal combination of CSP and Backtracking to optimally solve Sudoku in terms of execution time and memory consumption. Using these 4 algorithms as well as their optimal combination, we also would like to develop an application that would provide a friendly interface as well as the visualization of the execution of these algorithms to the user. In order to further improve the user convenience, instead of making the user type every digit of the sudoku (which would be very laborious), we would enable the user to just upload the sudoku by capturing its image from the camera, where the application would first convert the image into a 9x9 sudoku array for further processing. This would increase the user convenience and, therefore, name this project to **Sudoku Vision**.

II. Achievements

- Developed Application that solves Sudoku directly from Images
- Explored the solving of Sudoku using:
  - Backtracking
  - Optimised Backtracking with MRV Heuristic
  - Forward Checking
  - AC3
- Developed applications for each of the algorithm for good visualisation
- Compared Backtracking with Optimised Backtracking (time and memory)
- Performed execution time and memory consumption analysis to find the most efficient algorithm (as a mixture of Backtracking and CSP solving based methods)

III. Methodology

A. Digit Optical Charater Recogition (OCR)

This was done using a CNN-based model on the digit image dataset [1]. The exact details of the model architecture and dataset are given in the presentation as well as the Markdown Report.

B. Board Extraction (Image2Array)

The captured image was first converted to grayscale, followed by denoising and adaptive thresholding. After this, edge detection was performed, followed by contour analysis and cropping of the maximum contour. Then, the image was further divided into 9x9 parts followed by OCR, giving array. The exact details given in the presentation and Markdown Report.

C. Dataset

A total of 1500 sudokus were taken for analysis from [2]. They were classified into 3 levels of difficulties based on the number of zeros (empty locations): Hard: <50 zeros (267 Puzzles), Medium: >50 (873 Puzzles) and <55 zeros, Easy: >55 zeros (360 Puzzles)

D. Algorithms Used

In order to explore various methods of solving sudokus and finding the optimal combination of CSP and Backtracking algorithms, the following methods and their corresponding interactive applications for solving sudokus were explored. We will solve the sudoku (when using CSP solvers) by converting it into a CSP with row, column and box constraints, within the {1-9} domain. The methods are:

a. Backtracking: This is standard method for sudokus.

b. Optimised Backtracking (MRV Heuritics):
This is an optimised methodology of solving the backtracking method, where we use the Minimum Remaining Values (MRV) heuristic (it is also called most constrained variable or fail-first heuristic MRV). In this, we choose the variable with the fewest legal values, that is, pick a variable that is most likely to cause a failure soon.

c. Forward Checking: This is an algorithm for solving CSPs that prevents future conflicts by removing inconsistent values from the domains of unassigned variables that are dependent on the assigned variables.

d. AC3: This is an algorithm used in CSPs to ensure arc consistency by iteratively revising the domains of variables. It removes values from variable domains that do not satisfy constraints with connected variables, thus reducing the search space.

E. Algo Analysis (Optimal Combination)

In order to determine the optimal combination of the algorithms (Backtracking and Forward Checking), a detailed algorithmic analysis was carried out:

- For different difficulty levels of Sudokus
- For different number of iterations (of running Forward Checking Algorithm)
- The optimality of the algorithm was considered in terms of the “Mean Execution Time” and “Mean Memory Consumption” for 3 executions.

IV. Results

Algorithmic Analysis

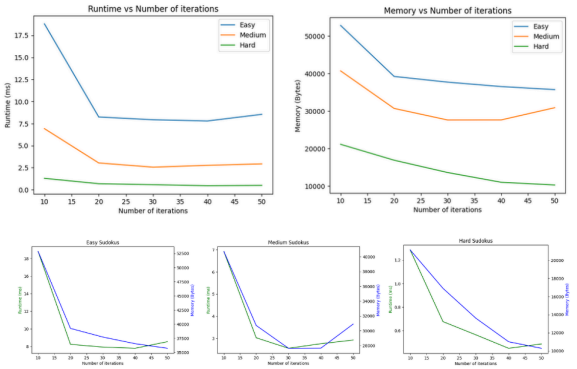

Fig. 1: Results from Algorithmic Analysis

Difficulty	Easy	Medium	Hard
n	~45	~30	~45

Table 1: Optimal n (Number of Iterations of Forward Check)

A detailed algorithmic analysis was carried out to find an optimal combination of Backtracking and Forward Checking algorithms in terms of memory consumption and execution time required. The results from this are displayed in Fig. 1 and Table. 1. The details are in the Markdown Report. An application was also developed from these results, which solved the sudoku in the most optimal way based on the difficulty level.

Applications

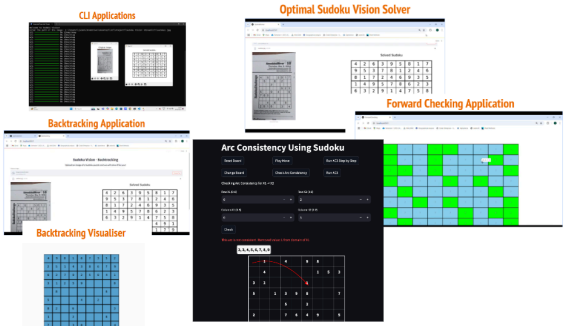

Fig. 2: Screenshots of Some Applications

In order to provide an interactive experience for the user to play sudoku and a good visualisation of the functioning of various applications, the above web-based applications were developed using Streamlit and Python. The following types of applications were developed:

- CLI Application: It is a simple command line application without any user interface, which takes in an image path and returns solved sudoku.
- Backtracking Application: This application allows the user to input a sudoku image and solve it. It also provides a visualisation of the execution of the algorithm on the sudoku.
- Optimised Backtracking Algorithm: This is similar to the previous application. However, it uses MRV heuristics-based variable selection, which leads to a different variable selection sequence, leading to a difference in the visualisation.
- Forward Checking Application: This application also solves sudoku directly from images using the forward checking algorithm. It also provides an interactive interface for the user to check and visualise the evolution of domains of any unassigned boxes at any iteration step, as selected by the user.
- AC3: This application explains how Arc Consistency Checks and AC3 algorithms work on sudokus. The users can visualise the step-by-step working of the AC3 algorithm along with the evolution of the domains. It also allows the users to visualise the impact of these arcs in the development of the constraints on the domains of the variables. The users can also set the values of the unassigned variables within the constraints and visualise the change in the domains of the other variables due to the presence of arcs.

These are very comprehensively built applications that provide solutions to the sudoku problem using a variety of methods with a good user interface and convenience.

V. Learnings and Conclusion

Through this project, we explored the methods of solving sudokus, starting with Backtracking. We also implemented the optimised version of backtracking using MRV Variable selection heuristics. The project also gave us an opportunity to analyse Sudoku as a Constraint Satisfaction Problem, leading us to use algorithms like Forward Checking and AC3. Apart from this, we also utilised various metrics to find the optimal combination of the algorithms, such as execution time and memory consumption. This project also gave us opportunities to improve our web development skills, namely in the Streamlit framework of Python, which gave us good experience in designing visualizations and applications. Finally, the determination to make the application user-centric led us to integrate this application with Computer Vision and Image Processing techniques to convert sudoku images directly to a sudoku array, preventing the users from performing the labour of manually typing every digit, finally yielding the name for the project, **Sudoku Vision**.

VI. References/Dataset Credits

[1] https://www.kaggle.com/datasets/kshitijdhama/printed-digits-dataset
[2] https://github.com/grantm/sudoku-exchange-puzzle-bank

CODE: https://github.com/JaidevSK/Sudoku-Vision
MARKDOWN REPORT: https://github.com/JaidevSK/Sudoku-Vision/blob/main/Report.md