Interactive AI Suite: Solving Puzzles and Playing Games with AI

Implementing Sudoku Solver, Tic-Tac-Toe, and Connect4 with Constraint Satisfaction and Game AI Techniques

Vraj Shah *22110292*
Shubham Agrawal *22110249*
Dewansh Singh Chandel *22110072*
Pranjal Gaur *22110201*
Parth Govale *22110087*
Yash Patkar *22110296*

***Abstract*—This report presents the implementation of an interactive suite comprising a Sudoku CSP solver, a Sudoku CSP helper, and two games: Tic-Tac-Toe and Connect4. Initially proposed with a dynamic visual interface and AI functionalities, the final implementation includes a Sudoku solver using the Constraint Satisfaction Problem (CSP) methodology, a helper tool for Sudoku that dynamically updates cell domains, and interactive gameplay for Tic-Tac-Toe and Connect4 with AI opponents. The project successfully demonstrates the application of AI algorithms in creating engaging and intelligent systems.**

I. Introduction

The project aims to integrate Artificial Intelligence principles into a suite of interactive applications. By developing a Sudoku CSP solver, a Sudoku CSP helper, Tic-Tac-Toe, and Connect4, we explored the practical applications of CSP algorithms and game AI strategies. The motivation stems from creating tools and games that are both functional and educational, showcasing AI's capabilities in solving structured problems and strategic gameplay.

II. Proposed Work

Our initial proposal included the following objectives:

- A Sudoku solver with a dynamic grid that allowed users to hover over cells to view possible values and updated the domains of other cells in real time based on input.
- Tic-Tac-Toe and Connect4 games which offer both traditional 2-player gameplay alongside AI opponents.

III. Implemented Work

The final implementation achieved the following:

A. Sudoku CSP Solver

The Sudoku CSP Solver is designed to solve Sudoku puzzles using a predefined grid. It employs Constraint Satisfaction Problem (CSP) algorithms, specifically leveraging a backtracking technique to ensure the puzzle is solved according to the strict rules of Sudoku. The solver iteratively fills the grid while checking for consistency, ensuring that each number placed adheres to the constraints imposed by the rows, columns, and subgrids.

- **Backtracking Algorithm**: The solver uses backtracking, a depth-first search approach, to explore potential solutions. For each empty cell, it attempts to place a valid number (1–9) from the available domain. If the value leads to a contradiction (violating Sudoku's constraints), the algorithm undoes the previous step and tries the next possible number. This process continues recursively until the puzzle is solved or all possibilities are exhausted.
- **CSP Constraints**: The solver maintains and enforces the CSP constraints at every step. These constraints are:
  - Each row must contain distinct numbers (1–9).
  - Each column must contain distinct numbers.
  - Each 3x3 subgrid must contain distinct numbers.

  As the solver progresses, it ensures that the current state of the grid does not violate these constraints, making it efficient in finding valid solutions.
- **Efficiency through Constraint Propagation**: While backtracking is the core of the solving process, constraint propagation helps reduce the search space. As the solver fills in cells, it updates the possible values (domains) for other cells dynamically, reducing the number of choices available for future decisions and increasing the efficiency of the backtracking process.

B. Sudoku CSP Helper

The Sudoku CSP Helper assists users by dynamically displaying possible values for empty cells based on the current grid. As users input values, the domains of all cells update in real time, reflecting the changes made according to the constraints of Sudoku.

- **CSP Representation**: Each cell is treated as a variable with a domain (1-9). The puzzle constraints are:
  - Distinct numbers in each row.
  - Distinct numbers in each column.
  - Distinct numbers in each 3x3 subgrid.

  These constraints define the CSP for Sudoku, with the tool updating domains based on user inputs and maintaining consistency across the grid.
- **Dynamic Domain Updates**: As users input a value in a cell, the tool eliminates that value from the domains of other cells in the same row, column, and 3x3 subgrid.

The domains of empty cells update in real time to reflect the current state of the grid, helping users make informed decisions.
- **Constraint Propagation**: The tool ensures that when a value is placed in a cell, it is removed from the domains of cells in the same row, column, and subgrid. This propagation reduces the search space and helps users avoid mistakes as they progress through the puzzle.

C. Tic-Tac-Toe

The Tic-Tac-Toe game offers two modes of gameplay:

- **2-player Traditional Gameplay**: In this mode, two human players take turns to place their marks (X or O) on the 3x3 grid. The objective is to get three of their marks in a row, column, or diagonal while blocking their opponent from doing the same. The game ends when either a player wins or the grid is completely filled, resulting in a draw.
- **Versus AI Mode**: The AI mode allows a player to compete against the computer. The AI uses the *Minimax* algorithm, a decision-making algorithm used to select the optimal move. Minimax evaluates all possible moves in the game tree, assuming that both players play optimally. The algorithm assigns a score to each move based on the potential outcomes: a positive score for a winning position, a negative score for a losing position, and a score of zero for a draw. The AI then chooses the move with the best score, ensuring the most strategic gameplay. The AI always plays second.

D. Connect4

The Connect4 game offers two modes of gameplay:

- **2-player Traditional Gameplay**: In this mode, two human players take turns dropping their discs into the columns of a 7x6 grid. The objective is to form a continuous line of four discs either horizontally, vertically, or diagonally. The game ends when a player achieves four discs in a row or when the grid is completely filled, resulting in a draw if no player wins.
- **Versus AI Mode**: In this mode, the player competes against the computer, which uses a *heuristic-based Minimax* algorithm with a depth of 3. The algorithm evaluates all possible moves in the game tree up to a depth of 3 and assigns scores to each move based on the expected outcome: a positive score for winning, a negative score for losing, and a score of zero for a draw. The AI selects the move with the highest score, assuming optimal play from both sides. The depth limit allows the AI to perform efficient decision-making without excessive computation. The AI always plays second.

Learning Outcomes

This project provided valuable insights into the application of Artificial Intelligence and Constraint Satisfaction Problems (CSP) in game and puzzle-solving contexts. The key learning outcomes include:

- **Understanding CSP Principles:** Developed a deep understanding of CSP techniques, including variable domains, constraint propagation, and backtracking algorithms, and their application in solving structured problems like Sudoku.
- **Minimax Algorithm and Heuristics:** Gained hands-on experience in implementing the Minimax algorithm and enhancing it with heuristics for AI gameplay in Tic-Tac-Toe and Connect4. Learned how to balance computational efficiency with strategic depth.
- **Real-Time Interactivity:** Acquired skills in designing real-time interactive tools, particularly in updating game states dynamically based on user actions and maintaining consistency across constraints.

IV. Conclusion and Future Work

The project demonstrates successful implementation of AI-based solutions in both problem-solving and interactive gameplay. Future enhancements could include:

- Allowing dynamic input for Sudoku puzzles rather than using a predefined matrix, or even creating our own random Sudoku generator and integrating it here.
- Adding advanced AI strategies for Tic-Tac-Toe and Connect4, such as Minimax with Alpha-Beta pruning, as well as adding a easy/medium/hard difficulty while playing with the AI player.
- Expanding the suite with additional games or puzzles leveraging AI.
- Enhancing the visualization and interactivity of the interface for improved user engagement.

Contributions

- **Vraj Shah and Shubham Agrawal:** Vraj and Shubham were primarily responsible for designing and implementing the *Sudoku CSP Solver* and *Sudoku CSP Helper*. They worked on creating the CSP framework, implementing the backtracking algorithm for the solver, and integrating constraint propagation for real-time domain updates in the helper.
- **Dewansh Singh Chandel and Pranjal Gaur:** Dewansh and Pranjal focused on the development of the *Tic-Tac-Toe* game, including the 2-player and versus AI modes. They implemented the Minimax algorithm for the AI mode, optimizing it to handle different game scenarios effectively.
- **Parth Govale and Yash Patkar:** Parth and Yash worked on implementing the *Connect4* game, handling both the 2-player mode and the AI mode. They integrated the heuristic-based Minimax algorithm with a depth of 3 to enable efficient and strategic gameplay for the AI.