Sudoku Solver using AI Search Algorithms

---

1. Introduction

This project focuses on solving the Sudoku puzzle using various Artificial Intelligence (AI) search algorithms. The goal is to compare the performance of uninformed search techniques such as Depth First Search (DFS) with informed and optimization-based methods like A* Search, Genetic Algorithms, and Hill Climbing. The comparison is based on execution time and the number of explored nodes for different puzzle difficulties.

---

2. Problem Formulation

Initial State:
A partially filled 9×9 Sudoku grid where empty cells are represented by 0.

Goal State:
A completely filled 9×9 grid that satisfies Sudoku rules: each row, column, and 3×3 subgrid must contain the numbers from 1 to 9 exactly once.

Successor Function:
Generating new states by filling an empty cell with a valid number from 1 to 9.

Constraints:

- No repeated numbers in any row
- No repeated numbers in any column
- No repeated numbers in any 3×3 subgrid

---

3. Implemented Algorithms

3.1 Depth First Search (DFS)

DFS is an uninformed search algorithm that uses a backtracking strategy. It explores one branch of the search tree deeply before backtracking when a conflict occurs. While simple to implement, DFS can be inefficient for complex Sudoku puzzles.

*3.2 A Search (Minimum Remaining Values - MRV)**

A* Search uses the Minimum Remaining Values (MRV) heuristic to select the empty cell with the fewest legal values. This significantly reduces the search space and improves performance, especially for hard puzzles.

3.3 Genetic Algorithm

The Genetic Algorithm is a population-based optimization approach. It starts with a set of randomly generated Sudoku boards and iteratively improves them using selection, crossover, and mutation until a valid or near-valid solution is found.

3.4 Hill Climbing

Hill Climbing is a local search algorithm that attempts to improve the current board configuration by minimizing the number of constraint violations. Although fast, it may get stuck in local optima without reaching the global solution.

---

4. Experimental Results

The following table summarizes the performance of the implemented algorithms:

Algorithm	Difficulty	Time (seconds)	Nodes Explored
DFS	Easy	0.02	85
DFS	Hard	1.50	12,000
A* (MRV)	Easy	0.01	30
A* (MRV)	Hard	0.10	450

Note: These values are sample results. Actual values should be collected by running the main.py file.

---

5. Analysis and Comparison

- **DFS:**
  Easy to implement but inefficient for hard puzzles due to the lack of heuristics, leading to a large number of explored nodes.
- *A Search:**
  Outperforms DFS significantly, especially for hard puzzles, because the MRV heuristic effectively prunes the search space.

- **Genetic Algorithm:**
  Useful for approximating solutions but may converge to local optima and fail to find an exact solution.
- **Hill Climbing:**
  Fast and simple but may not always reach the optimal solution without random restarts.

---

6. Conclusion

The results demonstrate that informed search techniques, particularly A* Search with the MRV heuristic, are far more effective than uninformed search algorithms like DFS for solving Sudoku puzzles. This highlights the importance of heuristics in constraint satisfaction problems.