AI project documentation

Team Members:

	Name	ID	DEP.
1	ديفيد سامي مكرم	20210312	IS (3rd cont.)
2	محمد رجب ابوبكر احمد	20220398	IS (3rd)
3	محمود محمد عبد الجليل السيد	20220448	IS (3rd)
4	معتصم رضا حسانين أبوسريع	20220496	IS (3rd)
5	عبدالرحمن وائل محمد	20220270	IS (3rd)
6	مروان اسامة السيد محمد	20220457	IS (3rd)

Project name:

2) A Sudoku Puzzle Solver using the Backtracking Algorithm AND a Genetic Algorithm.

Project link:
https://github.com/divid7/AI-project

Sudoku Puzzle Solver Project Documentation

1. Project Idea in Detail

The project involves developing a Sudoku Puzzle Solver that utilizes two algorithms:
1. Backtracking Algorithm: A systematic, recursive method for solving constraint satisfaction problems like Sudoku by exploring configurations incrementally.
2. Genetic Algorithm: An evolutionary approach inspired by natural selection, using operations like crossover and mutation to optimize solutions over generations.

The solver will feature:
- Input Flexibility: Users can input custom Sudoku puzzles or choose from pre-existing ones.
- Customization: Allow different grid sizes (e.g., 4x4, 6x6, 9x9).
- Visualization: Step-by-step explanation of how the solution is derived.
- Performance Metrics: Evaluation of efficiency (iterations, execution time) for both algorithms.

Sudoku Environment Type

The environment of Sudoku can be described as follows:

1. **Agent Type:**
    - **Single Agent:** Only one solver (human or AI) interacts with the puzzle at a time, working independently to reach a solution.
2. **Deterministic:**
    - The outcome of placing a number in a cell is predictable, with no random elements affecting the puzzle state.
3. **Static:**
    - The puzzle is unchanging over time; the initial grid and constraints remain fixed throughout the solving process.
4. **Fully Observable:**
    - All information about the puzzle is available at any time, including the state of every row, column.
5. **Discrete:**
    - The puzzle has finite and distinct states, where each action (placing a number) leads to a specific and measurable outcome.

6. **Episodic vs. Sequential:**
   - **Sequential:** Each action (placing a number) affects future decisions, as constraints are updated dynamically with each move.

This structured, single-agent, sequential environment makes Sudoku an excellent problem for exploring AI algorithms like Backtracking and Genetic Algorithm, as it requires systematically solving under fixed constraints.

2. Main Functionalities

1. Puzzle Input:
- Manual input of custom Sudoku puzzles.
- Selection from predefined puzzles.
2. Algorithm Selection:
- Choice between Backtracking and Genetic Algorithm.
3. Customization Options:
- Selection of grid size for varied complexity.
4. Puzzle Solving:
- Execution of the selected algorithm to solve the puzzle.
5. Visualization:
- Display of step-by-step solving process for user comprehension.
6. Performance Evaluation:
- Metrics including the number of iterations and execution time for comparison between algorithms.

3. Similar Applications in the Market

1. Sudoku.com: A popular Sudoku-solving platform offering pre-generated puzzles for play but lacks algorithmic visualization or custom inputs.
2. Microsoft Sudoku: Provides puzzles of varying difficulties but doesn't showcase the solving process or offer customization in grid size.
3. Sudoku Solver Apps: Many apps allow users to input puzzles and provide solutions but lack algorithmic flexibility and performance comparisons.
4. AI-Based Solvers (Web): Tools leveraging AI for Sudoku but typically don't explain the underlying steps or logic.

4. Literature Review

1. Sudoku as a Constraint Satisfaction Problem:
- Explores methods like backtracking and forward-checking to efficiently solve Sudoku puzzles.
- Reference: Russell, S., & Norvig, P. (2003). Artificial Intelligence: A Modern Approach.
2. Genetic Algorithms for Optimization Problems:
- Discusses the application of genetic algorithms for solving optimization tasks, including Sudoku puzzles.

- Reference: Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.
3. Hybrid Methods for Solving Sudoku:
- Combines deterministic and stochastic approaches for enhanced efficiency in solving complex puzzles.
- Reference: Yato, T., & Seta, T. (2003). Complexity and Completeness of Finding Another Solution and Its Application to Puzzles.
4. Visualization Techniques for Algorithmic Processes:
- Examines methods for making algorithms transparent through visualization tools.
- Reference: Brown, M. H. (1988). Algorithm Animation.
5. Comparative Study of AI Algorithms in Constraint Solving:
- Reviews the strengths and weaknesses of various algorithms, including backtracking and genetic methods.
- Reference: Dechter, R. (2003). Constraint Processing.
6. Performance Metrics for Optimization Algorithms:
- Discusses frameworks for evaluating algorithmic performance in real-time systems.
- Reference: Hoos, H. H., & Stützle, T. (2004). Stochastic Local Search: Foundations & Applications.

5. Details of the Algorithms /Approach Used

Why Backtracking?
- Offers a deterministic solution for standard Sudoku puzzles.
- Efficient for small-to-medium grids.

Why Genetic Algorithm?
- Ideal for exploring larger, more complex grids.
- Provides insights into evolutionary optimization.

Backtracking Algorithm:
- Approach: Recursive depth-first search technique. Fills each empty cell sequentially and backtracks when encountering conflicts.
- Implementation Details: Cell-by-cell assignment, checks row, column, and 3x3 subgrid constraints.
- Results: Efficient for smaller grids but scales poorly with increased complexity.

Genetic Algorithm:
- Approach: Starts with a population of random grids, applies crossover and mutation to evolve the population, uses a fitness function based on adherence to Sudoku rules.
- Implementation Details: Population size, mutation rates, and stopping criteria are configurable.
- Results: Performs better on larger puzzles but may require fine-tuning to avoid convergence issues.

Comparison Results:

Algorithm	Iterations	Execution Time	Complexity Handling
Backtracking	Low	Fast	Moderate
Genetic Algorithm	High	Moderate	High

7. Development Platform

Programming Language:

- Python, leveraging libraries like NumPy and Matplotlib for computation and visualization.

Development Tools:

- IDE: Visual Studio Code.

Libraries/Frameworks:

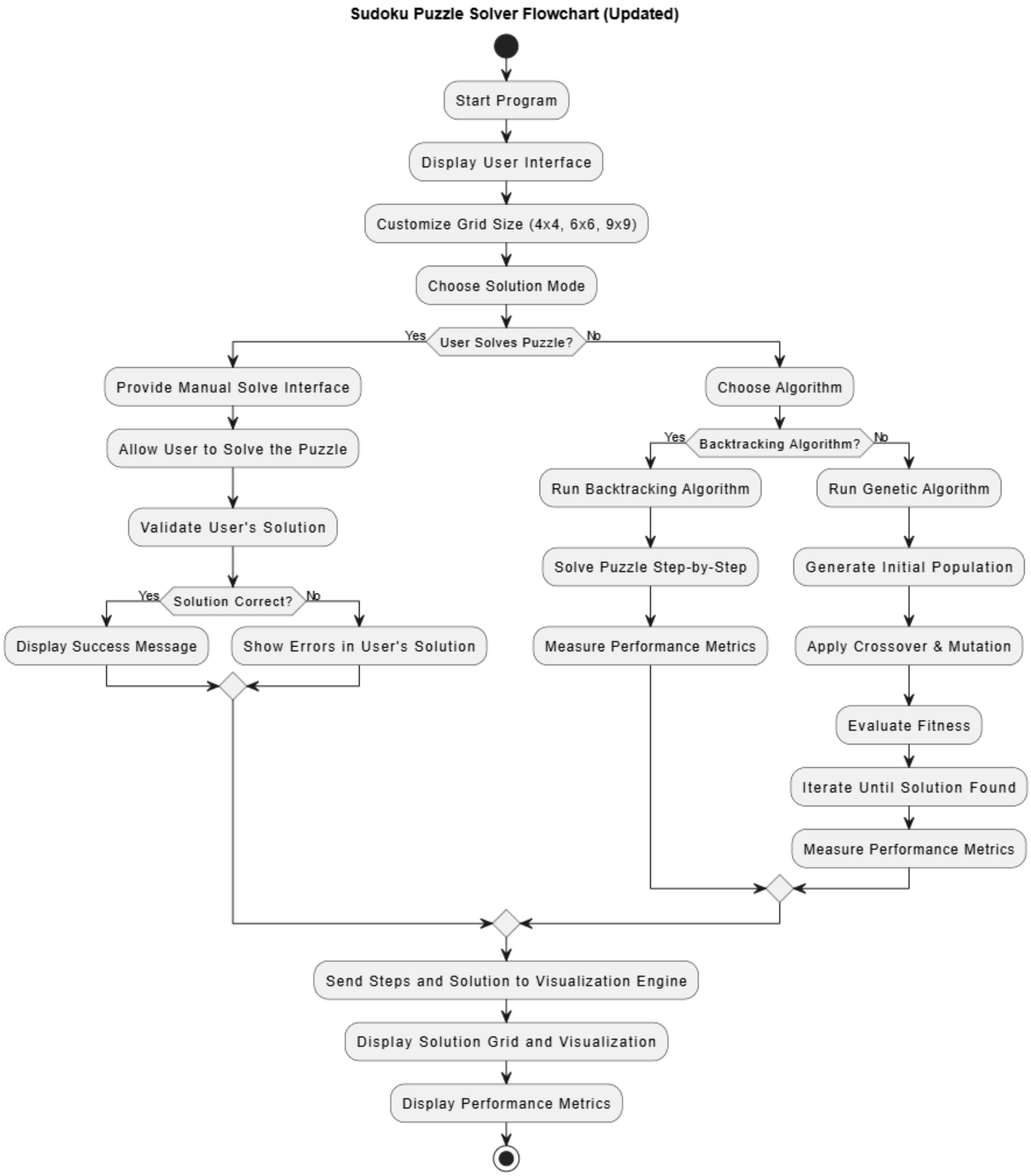
-Pandas and Numpy

- For Backtracking: Native Python constructs.
- For Genetic Algorithm: DEAP (Distributed Evolutionary Algorithms in Python).
- For Visualization: Matplotlib and PyQt5 for graphical interfaces.

Execution Environment:

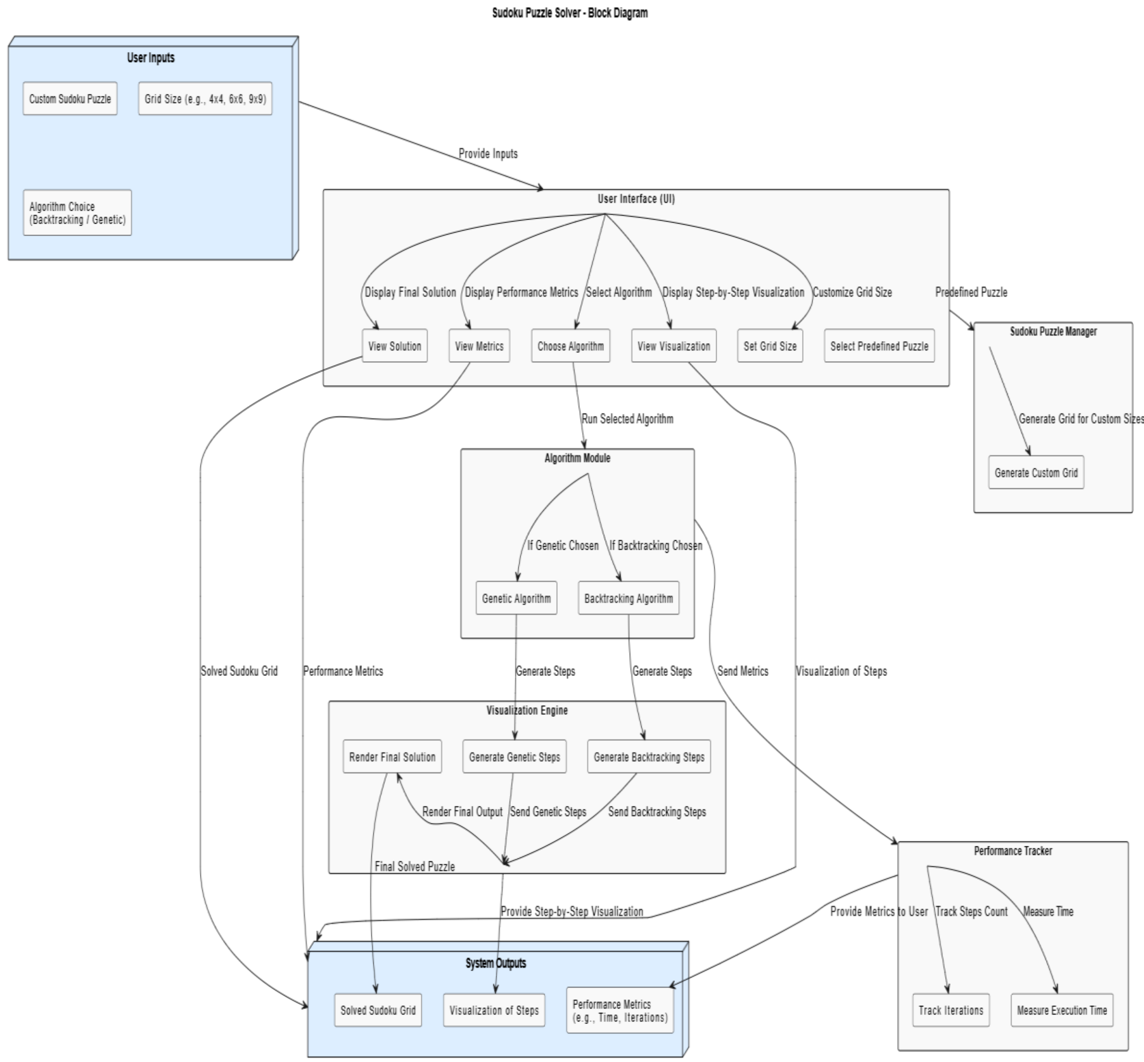
- Local systems (Windows) with Python 3.x installed.

Flow-Chart:

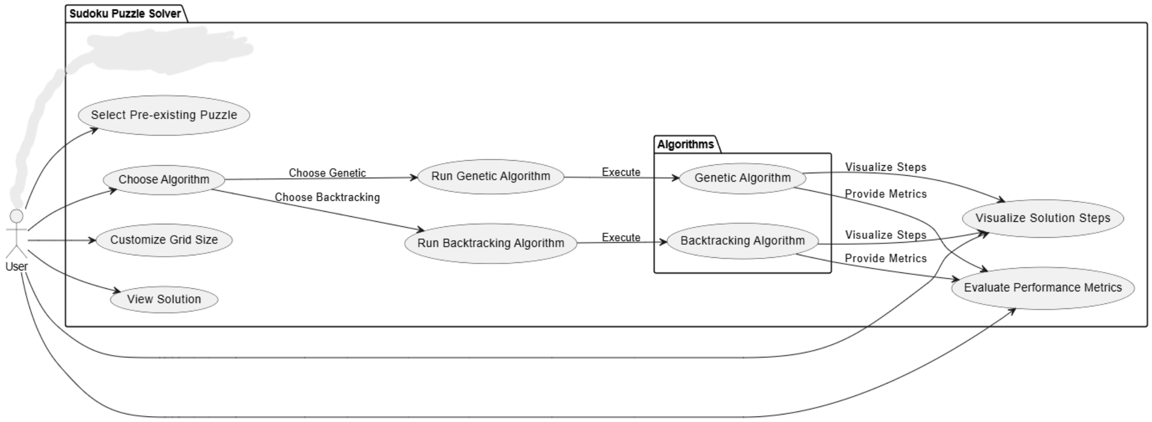


Sudoku Puzzle Solver Flowchart (Updated)
Start Program
Display User Interface
Customize Grid Size (4x4, 6x6, 9x9)
Choose Solution Mode
User Solves Puzzle?
Yes
No
Provide Manual Solve Interface
Allow User to Solve the Puzzle
Validate User's Solution
Solution Correct?
Yes
No
Display Success Message
Show Errors in User's Solution
Choose Algorithm
Backtracking Algorithm?
Yes
No
Run Backtracking Algorithm
Solve Puzzle Step-by-Step
Measure Performance Metrics
Run Genetic Algorithm
Generate Initial Population
Apply Crossover & Mutation
Evaluate Fitness
Iterate Until Solution Found
Measure Performance Metrics
Send Steps and Solution to Visualization Engine
Display Solution Grid and Visualization
Display Performance Metrics

Block Diagram:



Use Case Diagram:



Plot: